Who moved my pixels?!

User info

```
id
```

Mikhail Sosonkin

HTTP://DEBUGTRAP.COM

@HEXLOGIC

NYU

UNIVERSITY OF OXFORD

aws

VICE ON HBO
I'm on TV!!

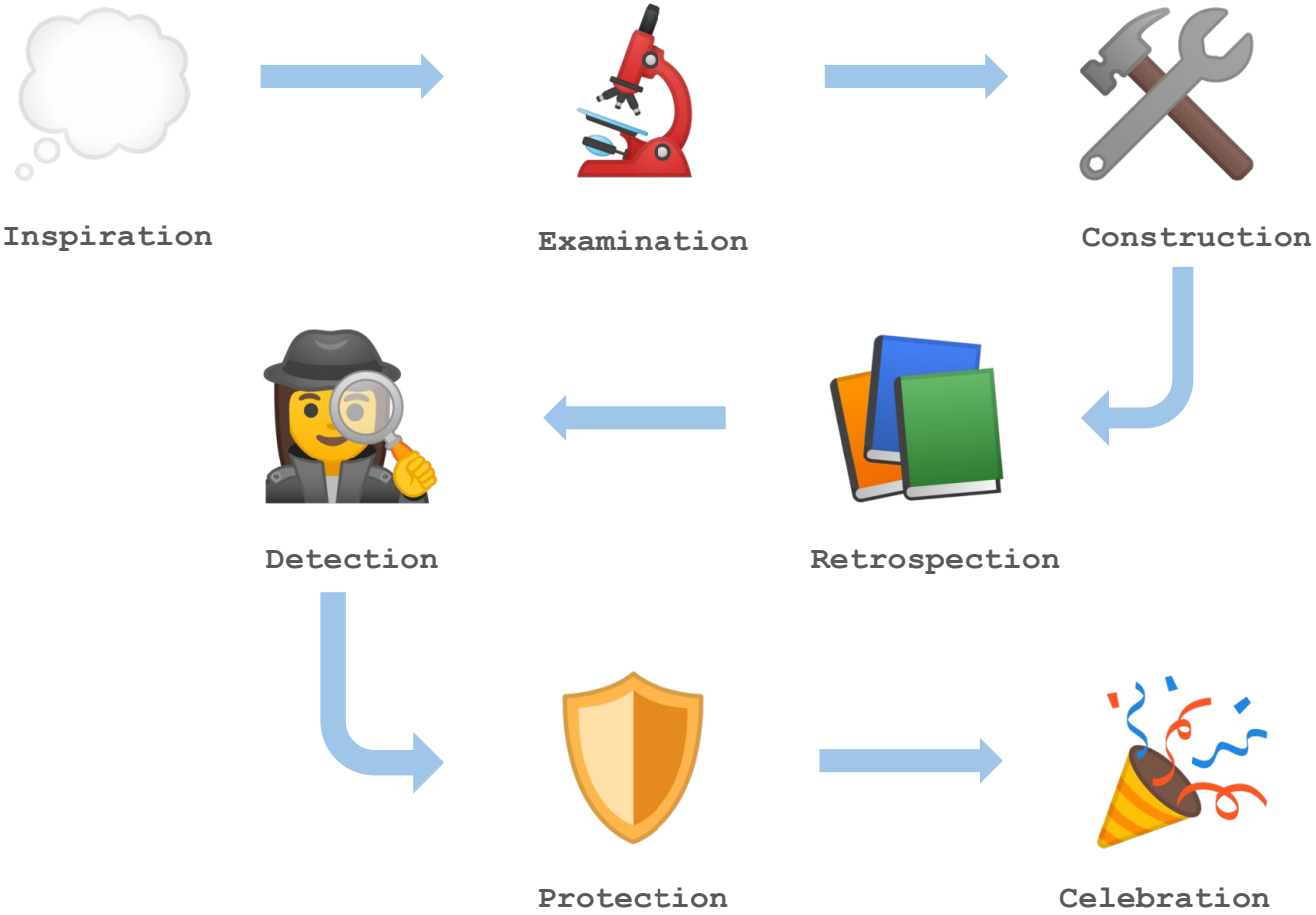
“Synack leverages the best combination of humans and technology to discover security vulnerabilities in our customers’ web apps, mobile apps, IoT devices and infrastructure endpoints”

Why malware?

Motives and intent

- Pranking
- Hacktivism
- Ransom
- Compute theft
- Espionage
- Financial theft
- *For educational purposes!*

outline


Inspiration
Examination
Construction
Retrospection
Detection
Protection
Celebration

ZooPark
the inspiration

- Reported by Kaspersky in May 2018
- Android based malware
- Targeted middle eastern phones
- Focuses on information theft
  - Exfiltration via HTTP GET request parameters
- Distributed via Watering Holes
  - Alnaharegypt[.]com - with an iframe to malicious APK.

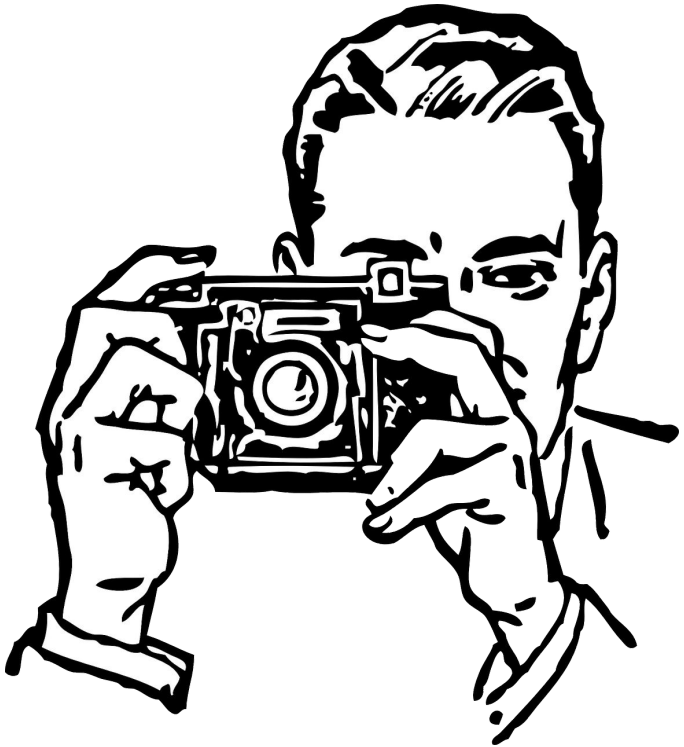
ZooPark
features

- *Version 1*
  - Contacts/Accounts
- *Version 2*
  - Call logs/GPS Location
  - SMS Messages/Device Information
- *Version 3*
  - Audio Call Records
  - Installed Application Details
  - Browser Data - bookmarks & History
  - Photos & Pictures from memory card
- *Version 4 (suspected to be outsourced)*
  - Keylogs/Clip data
  - Arbitrary file/folders
  - Capturing photos/videos/audio/ **screenshots**/screen records
  - External Application data (Telegram, WhatsApp, IMO, Chrome)
  - Remote Shell/Silently Sending SMS/Making phone calls

F536BC5B79C16E9A84546C2049E810E1
Android screencapture malware in not unique! (July 2017)

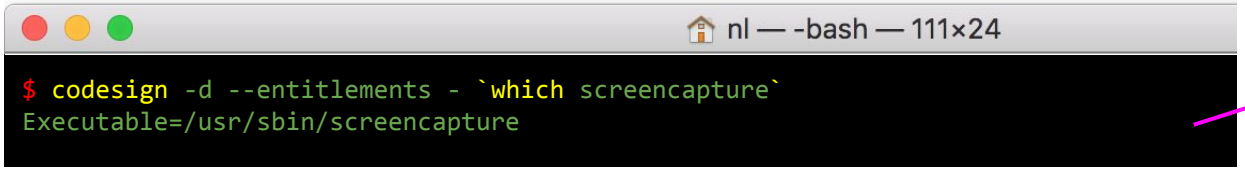
How do you capture screens?

on OS X



Which screencapture

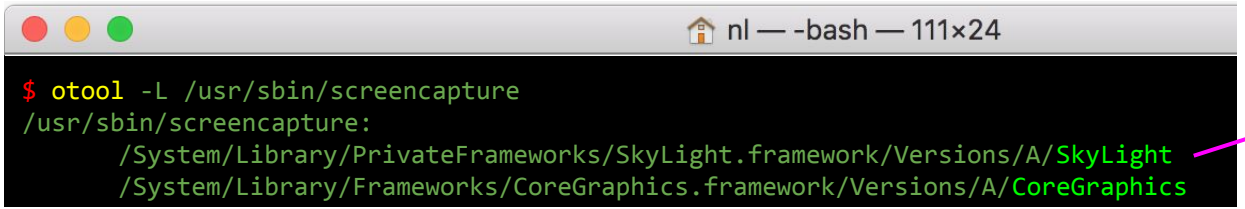
an overview



No Special entitlements!

The "bin" version contains general-use programs, while the "sbin" version contains programs that're generally only used for system administration.

http://www.westwind.com/reference/OS-X/invisibles.html



Shared with WindowServer!

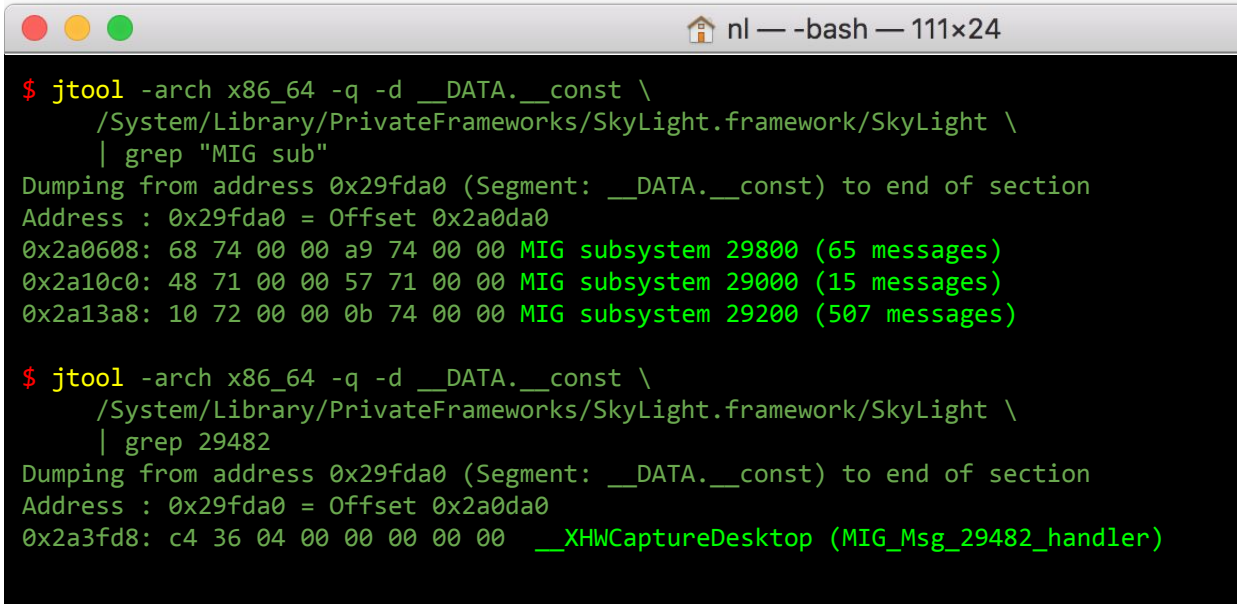
Libraries with the relevant functions:

_CGDisplayCreateImage (CoreGraphics)
-> **_SLSHWCaptureDesktop** (SkyLight)

Maps to **_XHWCaptureDesktop** in server-side SkyLight

Mach based IPC

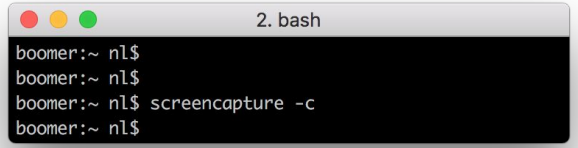
mig

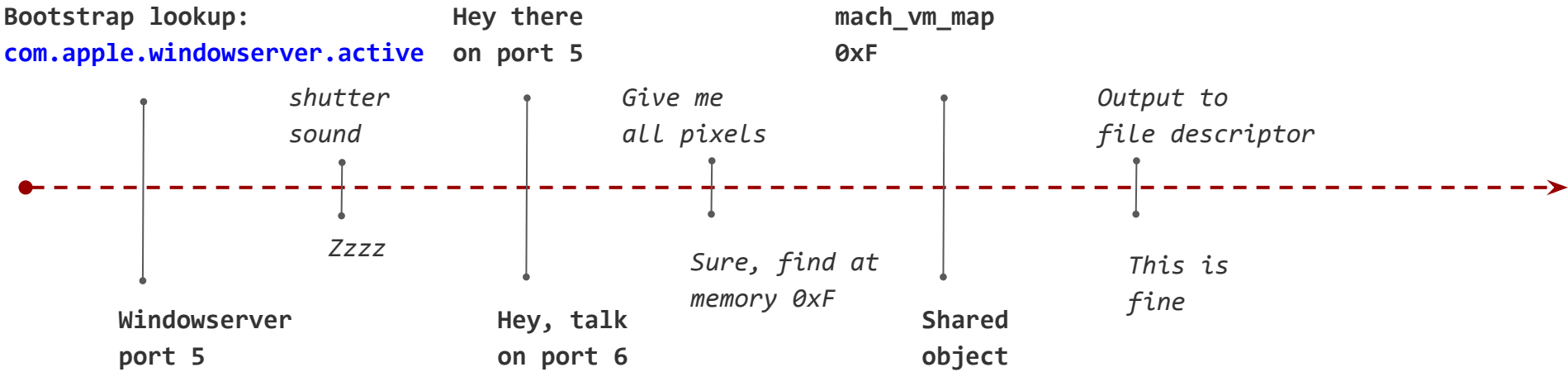


Mach 3 Server Writer's Guide:
http://shakthimaan.com/downloads/hurd/server_writer.pdf

Highlevel

the utility





Launchd

WindowServer

More details: http://debugtrap.com/2018/04/01/screencapure/

Into the bytes

1) Find ports

```
__text:34A6 _CGSLookupServerRootPort proc near
                ; CODE XREF: _get_session_port+1Fp
                ; _CGSLookupServerPort+A5p ...
__text:34A6 lookup_result   = dword ptr -38h
__text:34A6 special_port    = dword ptr -34h
...
__text:34C8 lea     rdx, [rbp+special_port] ; special_port
__text:34CC mov     dword ptr [rdx], 0
__text:34D2 mov     r13, cs:_mach_task_self__ptr
__text:34D9 mov     edi, [r13+0]    ; task
__text:34DD mov     esi, 4          ; which_port
__text:34E2 call    _task_get_special_port
__text:34E7 mov     ebx, eax
__text:34E9 test    ebx, ebx
__text:34EB jnz     loc_35ED
__text:34F1 mov     edi, [rbp+special_port]
__text:34F4 test    edi, edi
__text:34F6 jz      loc_35ED
__text:34FC lea     rdx, [rbp+lookup_result]
__text:3500 mov     dword ptr [rdx], 0
__text:3506 lea     r14, aCom_apple_w_21
            ; "com.apple.windowserver.active"
__text:350D xor     ecx, ecx
__text:350F mov     r8d, 8
__text:3515 mov     rsi, r14
__text:3518 call    _bootstrap_look_up2
```

```
mach_port_t self = mach_task_self();
task_get_bootstrap_port(self, &bs_port);

// find the server port
if(bootstrap_look_up(bs_port,
      "com.apple.windowserver.active",
      &serv_port) == KERN_SUCCESS) {
    printf("Server port: 0x%08X\n",
                serv_port);
} else {
    printf("Error looking up server"
           " port\n");
    exit(2);
}
```

Into the bytes

2) make session

```
__text:5394 __CGSNewConnectionPort proc near
                ; CODE XREF: _SLSNewConnection+1DDp
                ; _SLSNewConnection+240p
...
__text:5430 mov     [rbp+msg.msgh_bits], 80001513h
__text:543A mov     [rbp+msg.msgh_remote_port], r12d
__text:5441 call    _mig_get_reply_port
__text:5446 mov     [rbp+msg.msgh_local_port], eax
__text:544C mov     [rbp+msg.msgh_id], 7468h
__text:5456 mov     [rbp+msg.msgh_reserved], 0
__text:5460 cmp     cs:_voucher_mach_msg_set_ptr, 0
__text:5468 jz      short loc_547D
__text:546A lea     r14, [rbp+msg]
__text:5471 mov     rdi, r14
__text:5474 call    _voucher_mach_msg_set
__text:5479 mov     eax, [r14+0Ch]
__text:547D loc_547D:
__text:547D mov     [rsp+480h+notify], 0 ; notify
__text:5484 lea     rdi, [rbp+msg]  ; msg
__text:548B mov     esi, 3          ; option
__text:5490 mov     ecx, 3Ch ; '<'  ; rcv_size
__text:5495 xor     r9d, r9d        ; timeout
__text:5498 mov     r8d, eax        ; rcv_name
__text:549B mov     edx, ebx        ; send_size
__text:549D call    _mach_msg
```

```
//   __CGSNewConnectionPort
mach_msg_header_t* req_port =
                    (mach_msg_header_t*)buffer;
req_port->msgh_bits = 0x80001513;
req_port->msgh_size = 1;
req_port->msgh_remote_port = serv_port;
req_port->msgh_local_port =
mig_get_reply_port();
req_port->msgh_voucher_port = 0;
req_port->msgh_id = 0x7468;

uint32_t* values = (uint32_t*)
          (buffer + sizeof(mach_msg_header_t));
values[1] = port_right;

// set the voucher
voucher_mach_msg_set(req_port);
mach_msg_return_t ret = mach_msg(req_port,
      0x3, 0x48, 60,
      req_port->msgh_local_port, 0, 0);
mach_port_t session_port =
       ( (mach_port_t*)buffer )[7];
```

Into the bytes

3) get pixels

```
__text:1F796C _SLSHWCaptureDesktop proc near
                        ; CODE XREF: _SLDisplayCreateImage+1Dp
                        ; _SLDisplayCreateImageForRect+F7p
...
__text:1F8364 mov     dword ptr [rbp+msg_body10_var_80+4], eax
__text:1F8367 mov     [rbp+msg.msgh_bits], 1513h
__text:1F8371 mov     eax, [rbp+remote_port]
__text:1F8377 mov     [rbp+msg.msgh_remote_port], eax
__text:1F837D call    _mig_get_reply_port
__text:1F8382 mov     [rbp+msg.msgh_local_port], eax
__text:1F8388 mov     [rbp+msg.msgh_id], 732Ah
__text:1F8392 mov     [rbp+msg.msgh_reserved], 0
__text:1F839C cmp     cs:_voucher_mach_msg_set_ptr , 0
__text:1F83A4 jz      short loc_1F83B8
__text:1F83A6 lea     rdi, [rbp+msg]
__text:1F83AD call    _voucher_mach_msg_set
__text:1F83B2 mov     eax, [rbp+msg.msgh_local_port]
__text:1F83B8 loc_1F83B8 :
__text:1F83B8 sub     rsp, 8
__text:1F83BC mov     esi, 3            ; option
__text:1F83C1 mov     edx, 72           ; send_size
__text:1F83C6 mov     ecx, 136          ; rcv_size
__text:1F83CB xor     r9d, r9d          ; timeout
__text:1F83CE lea     rdi, [rbp+msg]    ; msg
__text:1F83D5 mov     r8d, eax          ; rcv_name
__text:1F83D8 push    0                 ; notify
__text:1F83DA call    _mach_msg
```

```
struct req_msg* rq_msg = (struct req_msg*)buffer;

// ...
rq_msg->header.msgh_id = 0x732A;

// x, y, width, height
rq_msg->x = 0.0;
rq_msg->y = 0.0;
rq_msg->width = 1024.0;
rq_msg->height = 768.0;

// display id values (vm: 0x5b81c5c0, non-vm: 0x042499b0)
rq_msg->display_id = 0x042499b0;
rq_msg->param5 = 0x00000441;

// set the voucher
voucher_mach_msg_set(&rq_msg->header);

if(mach_msg(&rq_msg->header, 0x3, 0x48, 0x88,
      rq_msg->header.msgh_local_port, 0, 0) !=
                                  MACH_MSG_SUCCESS) {}

mig_put_reply_port(rq_msg->header.msgh_local_port);

uint32_t object = *(uint32_t*)
  ( ((u_char*)buffer) + sizeof(mach_msg_header_t) + 4);
if(mach_vm_map(self, rq_msg, size, 0, 0x1, object,
              0x0, false, 3, 3, true) != KERN_SUCCESS) { }
```

Putting it all together

Write better malware

List all displays

Encode and write image to file

Create image from display

No Shutter sound (-x)!

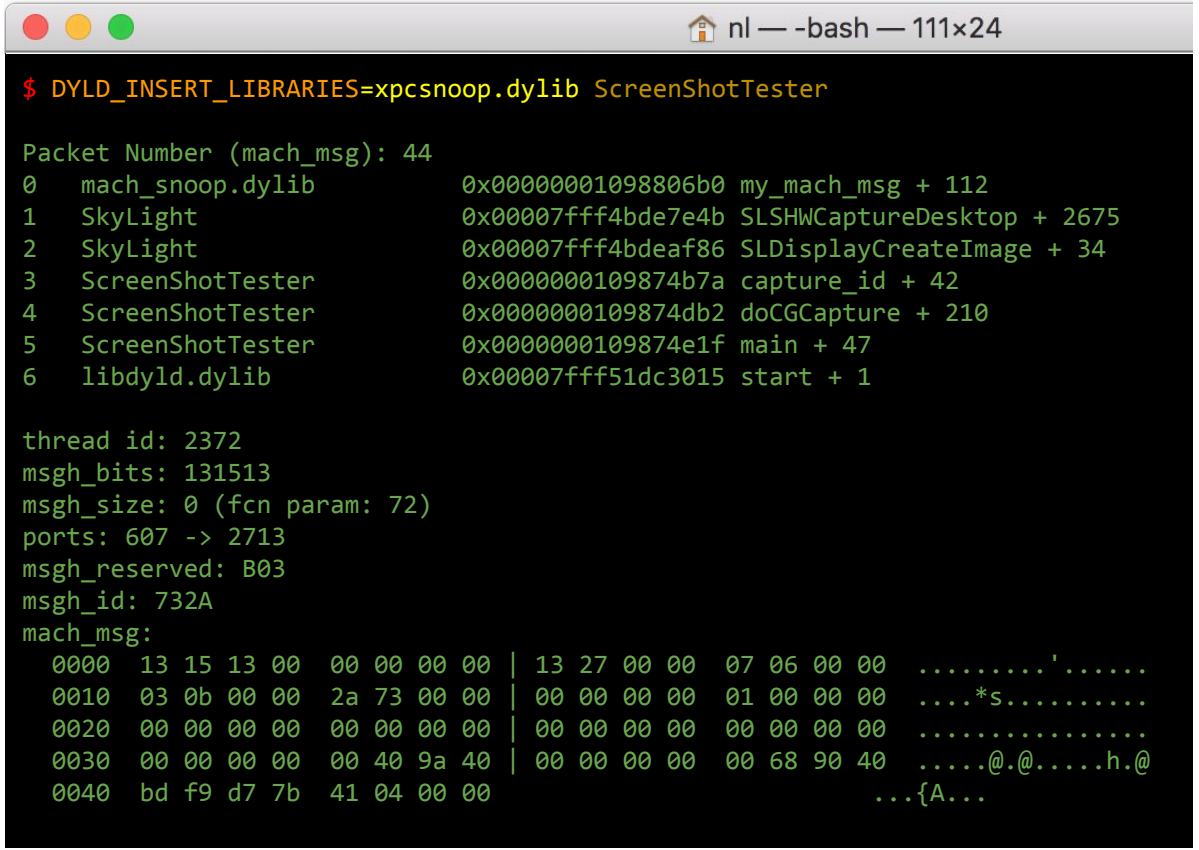
```
void doCGCapture() {
    CGDirectDisplayID displays[256];
    uint32_t dispCount = 0;

    CGGetActiveDisplayList(256, displays, &dispCount);

    for(int i = 0; i < dispCount; i++) {
        CGDirectDisplayID dispId  = displays[i];

        CGImageRef img = CGDisplayCreateImage(dispId);

        char path_str[1024];
        snprintf(path_str, 1023, "./image%d.png", i);

        CFURLRef path =
            CFURLCreateWithFileSystemPath (NULL,
                    __CFStringMakeConstantString (path_str),
                    kCFURLPOSIXPathStyle , false);

        CGImageDestinationRef destination =
            CGImageDestinationCreateWithURL(
                    path, CFSTR("public.png"), 1, NULL);

        CGImageDestinationAddImage(destination, img, nil);
        CGImageDestinationFinalize(destination);
    }
}
```

Sniffing mach

via injection

Inject and intercept mach related function calls.

Preferably on a small set of API's due to volume of messages




```
static const interpose_t
 interposing_functions[] __attribute__
      ((used,section("__DATA, __interpose"))) = {
 { (void*) my_bootstrap_look_up2, (void*)bootstrap_look_up2 },
 { (void*) my_mach_msg,           (void*)mach_msg },
 { (void*) my_mach_msg_overwrite, (void*)mach_msg_overwrite }};
```

Mach msg sequence

MIG

1. `msgh_id = 0x7152 // start a session`
2. `msgh_id = 0x7148 // test connection, get MIG version`
3. `msgh_id = 0x7475 // display state, find the display ID`
4. `msgh_id = 0x7468 // create secondary session`
5. `msgh_id = 0x732A // request screen pixels`
6. `mach_vm_map      // map pixels to local process memory`

There is more than one way

There usually is!

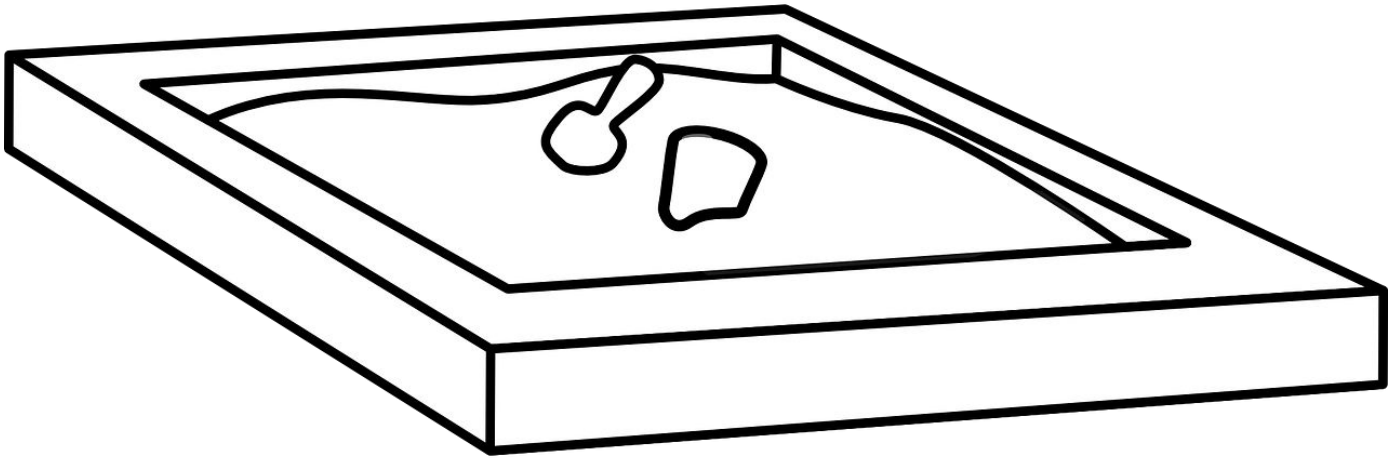
```
CGImageRef screenshot = CGWindowListCreateImage(
  CGRectInfinite,
  kCGWindowListOptionOnScreenOnly,
  kCGNullWindowID,
  kCGWindowImageDefault);

NSBitmapImageRep *bitmapRep =
     [[NSBitmapImageRep alloc] initWithCGImage:screenshot];
```

rdar://37423927

https://krausefx.com/blog/mac-privacy-sandboxed-mac-apps-can-take-screenshots

The sandbox will protect me

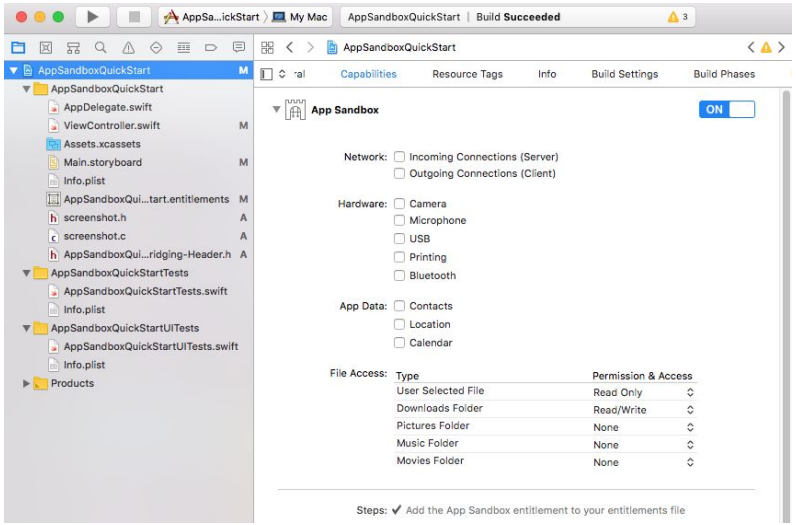
sometimes



Malware in action

video demo

1. Cocoa base application
2. Runs in a full sandbox
3. File write allowed for demo



Sandbox

Chrome

```
(allow mach-lookup (global-name "com.apple.windowserver.active"))
```

- cdm.sb:
  - Content Decryption Module (CDM). A CDM is required for a media renderer, audio decoder or video decoder to handle protected content (chromium).
- gpu_v2.sb
  - Allow communication between the GPU process and the UI server.
- ppapi_v2.sb
  - Pepper Plugin API (PPAPI) was initially only supported by Google Chrome and Chromium. Later, other Chromium-based browsers such as Opera and Vivaldi, also added PPAPI plugin support.

https://www.chromium.org/developers/design-documents/sandbox/osx-sandboxing-design

Sandbox

firefox

```
static const char widevinePluginSandboxRulesAddend[] = R"SANDBOX_LITERAL(
  (allow mach-lookup (global-name "com.apple.windowserver.active"))
) SANDBOX_LITERAL";
```

- **Widevine DRM is also used with the Chromium web browser and on Android.**

```
// The "Safe Mode" Flash NPAPI plugin process profile
static const char flashPluginSandboxRules[] = R"SANDBOX_LITERAL(
 ; Services
  (allow mach-lookup
      (global-name "com.apple.windowserver.active")
```

```
static const char contentSandboxRules[] = R"SANDBOX_LITERAL(
  (version 1)
  (if (string=? hasWindowServer "TRUE")
    (allow mach-lookup (global-name "com.apple.windowserver.active")))
```

https://hg.mozilla.org/mozilla-central/file/tip/security/sandbox/mac/SandboxPolicies.h

Sandbox

Positron - a experimental, Electron-compatible runtime on top of Gecko

```
static const char widevinePluginSandboxRulesAddend[] =
 "(allow mach-lookup (global-name
\"com.apple.windowserver.active\"))\n";
```

- **Widevine DRM is also used with the Chromium web browser and on Android.**

```
static const char contentSandboxRules[] =
 "(version 1)\n"
 "\n"
 "(define sandbox-level %d)\n"
 "  (allow mach-lookup\n"
 "      (global-name
\"com.apple.windowserver.active\")\n"
```

https://github.com/mozilla/positron/blob/master/security/sandbox/mac/Sandbox.mm

Sandbox

Safari - Webkit

com.apple.WebProcess.sb.in:

```
(allow mach-lookup
#if __MAC_OS_X_VERSION_MIN_REQUIRED >= 101400
#else
      (global-name "com.apple.windowserver.active")
#endif
)
#if __MAC_OS_X_VERSION_MIN_REQUIRED >= 101400
(deny mach-lookup (with no-log)
   (global-name "com.apple.windowserver.active"))
#endif
```

com.apple.WebKit.plugin-common.sb.in:

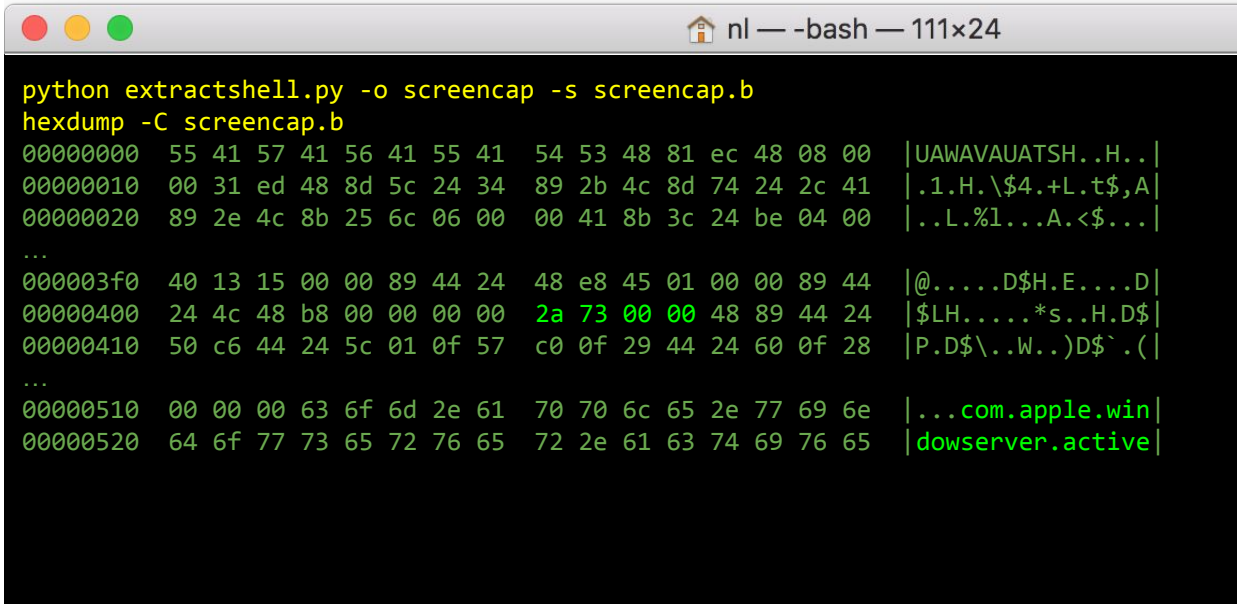
```
;; Various services required by AppKit
;; and other frameworks
(allow mach-lookup
   (global-name "com.apple.windowserver.active")
```

Window server gets abused a lot:
https://blog.ret2.io/2018/07/25/pwn2own-2018-safari-sandbox/

https://github.com/WebKit/webkit

Why is that an issue?

Protected by sandbox



Shellcode can do it in about 9076 bytes

mach_msg and **mach_vm_map** are mach system calls

Can networking and windowserver be accessed at the same time?

https://github.com/nologic/shellcc **SHELLCC is a build environment for making shellcode in C using GCC and other binary tool. It is meant to enable people with limited knowledge of assembly to write quality shellcode as well as bring code maintainability by using a high level language.**

Efficient collection

Thanks touchbar

- **Screenshot of regular resolution is about: 1.4M PNG Compressed.**
- **An image of touchbar can be used for a more efficient selection of timing.**

Chrome is active:

Sublime has these documents open:

And now, the terminal:

What malware is out there?

... in the wild

Querying for screencapture capability

on VirusTotal

```
rule combined_screen_cap {
    strings:
        $utility = "/usr/sbin/screencapture" nocase

        $lib = "CoreGraphics"
        $func = "CGDisplayCreateImage"

        $mov_msg_id = { C7 ?? 54 FF FF FF 2A 73 00 00 }
        $mach_msg = "_mach_msg"

    condition:
        $utility or
        ($lib and $func) or
        ($mov_msg_id and $mach_msg)
}
```

Basic triage*

signatures

Static analysis

disassembly

Static analysis

partial emulation

Dynamic analysis

Sandboxing

Dynamic analysis

Emulation

Manual analysis

APT!

Finding malware

thanks VT/Yara

1124 matched samples
15 Anti-virus (AV) flagged
6 Sample with great than 1 AV

Great for research, need better rules for anything production

Simple static signatures aren't great for endpoint protection

Filtering samples

Getting around throttling

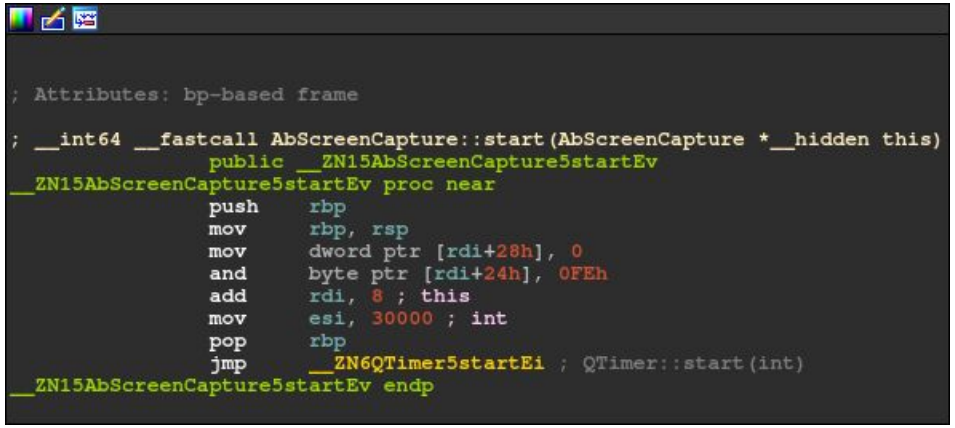
```
while [ -z "$DONE" ]; do
    DONE="done"

    for hash in `cat hashes`; do
        if [ -z "`cat $hash.json`" ]; then
            curl -s --request POST \
              --url 'https://www.virustotal.com/vtapi/v2/file/report' \
              -d apikey=${APIKEY} \
              -d "resource=$hash" > $hash.json

            cat $hash.json
            sleep 2
            DONE=
        fi
    done
    sleep 2
done
```

Meeting the malware: Mokes

664e0a048f61a76145b55d1f1a5714606953d69edccec5228017eb546049dc8c

Mokes

- **Uses CoreGraphics via Qt**
- Specifically:
  QCocoaScreen::grabWindow
- Capturing Screen (every 30 sec.)



```
; Attributes: bp-based frame

; __int64 __fastcall AbScreenCapture::start(AbScreenCapture *__hidden this)
                public __ZN15AbScreenCapture5startEv
__ZN15AbScreenCapture5startEv proc near
                push    rbp
                mov     rbp, rsp
                mov     dword ptr [rdi+28h], 0
                and     byte ptr [rdi+24h], 0FEh
                add     rdi, 8 ; this
                mov     esi, 30000 ; int
                pop     rbp
                jmp     __ZN6QTimer5startEi ; QTimer::start(int)
__ZN15AbScreenCapture5startEv endp
```

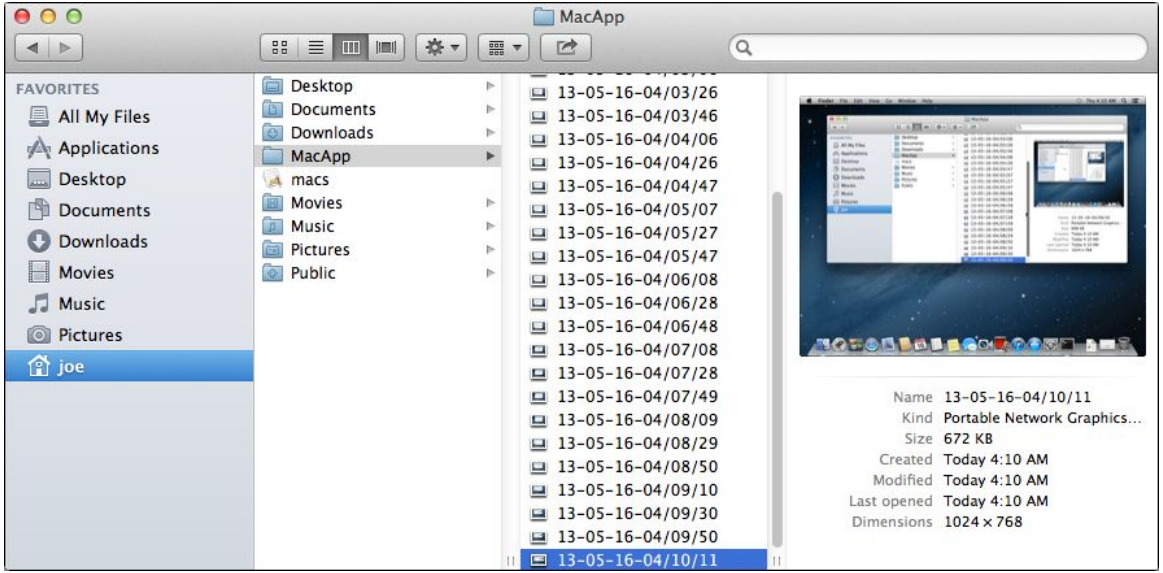
```
QPixmap QCocoaScreen::grabWindow(WId window,
    int x, int y, int width, int height) const
{

    // 128 displays should be enough
    //  for everyone.
    const int maxDisplays = 128;
    CGDirectDisplayID displays[maxDisplays];
    CGDisplayCount displayCount;
    ...

    for (uint i = 0; i < displayCount; ++i) {
        const CGRect bounds =
              CGDisplayBounds(displays[i]);
        ...
        QCFType<CGImageRef> cgImage =

CGDisplayCreateImage(displays[i]);
        const QImage image =
              qt_mac_toQImage(cgImage);
    }
}
```

Meeting the malware: Macs

6acd92d0dfe3e298d73b78a3dcc6d52ff4f85a70a9f2d0dcfe7ae4af2dd685cc

[Macs]

- **Uses '/usr/sbin/screencapture -x [] -T 20'**
- **It dumps screenshots into a folder called MacApp**



Meeting the malware: blackhole

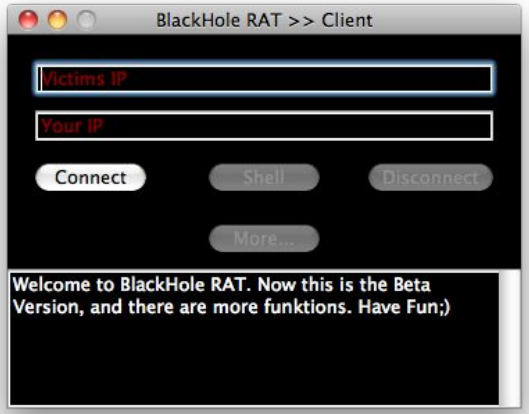
d1a0c589b3ac33626d47156aaa938be92ba3d6f889de5a6f62d4afe0ac2cf65a

blackhole

- Comes with a PPC version
  - why?!

- **/usr/sbin/screencapture -x **
  /Applications/.JavaUpdater/.Data/Screen.png

- No one reports on screen capture capability



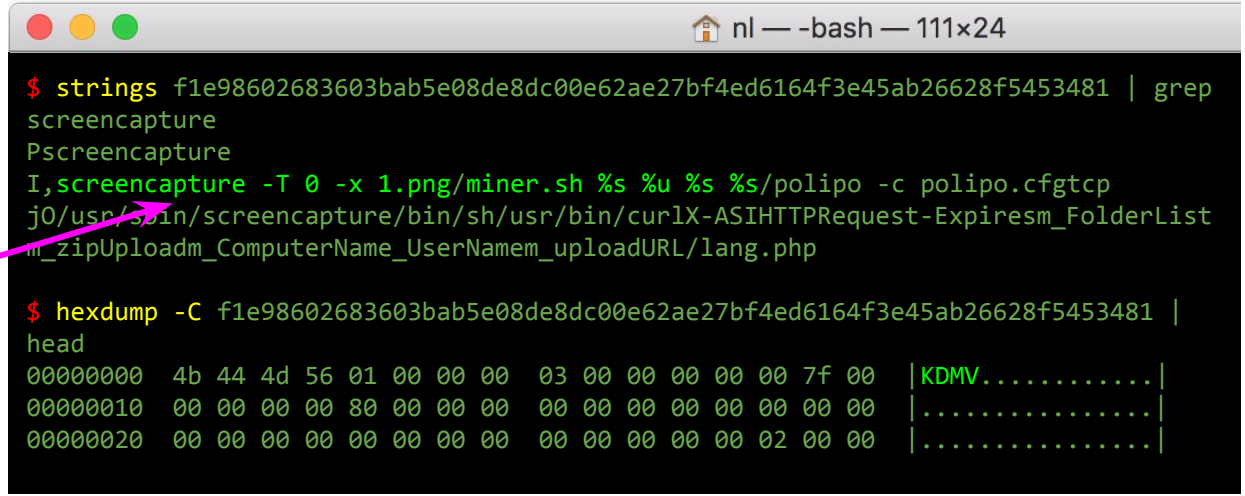
Meeting the malware: Eh?!

f1e98602683603bab5e08de8dc00e62ae27bf4ed6164f3e45ab26628f5453481

vmdk part with malware

- I'm not malware
- Why would someone upload this to VT?
- References screencapture but nothing directly executable

Definitely some kind of malware in there



Meeting the malware: WINDSHIFT

Need a hash

Recent findings by DarkMatter at HiTB

APT? 

Current Tool-set by chronological order, mostly cyber espionage tools, still under on-going development:

Dark Matter Code	Target OS	First seen	Description
WINDTAIL.A	macOS	Jan - 2017	Backdoor exfiltrating files
WINDTAIL.B	macOS	Jan - 2018	Downloader of WINDTAPE
WINDTAIL.C	macOS	Jan - 2018	Variant of WINDTAIL.B
WINDTAPE	macOS	Jan - 2018	Backdoor taking screenshots
WINDDROP - unconfirmed	Windows	May - 2018	Downloader of a unknown malware

How do we find the behavior?

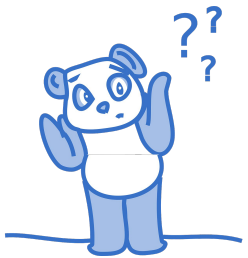
on OS X

On the endpoint

file attributes

```
nl — -bash — 111×24

$ mdfind kMDItemIsScreenCapture = 1
/Users/nl/Desktop/Screen Shot 2018-05-15 at 10.52.35 PM.png
/Users/nl/Desktop/Screen Shot 2018-04-24 at 6.26.21 PM.png
/Users/nl/Desktop/Screen Shot 2018-04-24 at 6.25.52 PM.png

$ ls -l@ "Screen Shot 2018-05-15 at 10.52.35 PM.png"
-rw-r--r--@ 1 nl  staff  120087 May 15 22:52 /Users/nl/Desktop/Screen Shot
2018-05-15 at 10.52.35 PM.png
        com.apple.FinderInfo          32
        com.apple.lastuseddate#PS     16
        com.apple.metadata:kMDItemIsScreenCapture         42
        com.apple.metadata:kMDItemScreenCaptureGlobalRect         86
        com.apple.metadata:kMDItemScreenCaptureType       51

$ mdls "Screen Shot 2018-05-15 at 10.52.35 PM.png"
_kMDItemDisplayNameWithExtensions   = "Screen Shot 2018-05-15 at 10.52.35 PM.png"
kMDItemIsScreenCapture              = 1

$ xattr -l "Screen Shot 2018-05-15 at 10.52.35 PM.png"
com.apple.metadata:kMDItemIsScreenCapture:
00000000  62 70 6C 69 73 74 30 30 09 08 00 00 00 00 00 00  |bplist00........|
00000010  01 01 00 00 00 00 00 00 00 01 00 00 00 00 00 00  |................|
00000020  00 00 00 00 00 00 00 00 00 09
```

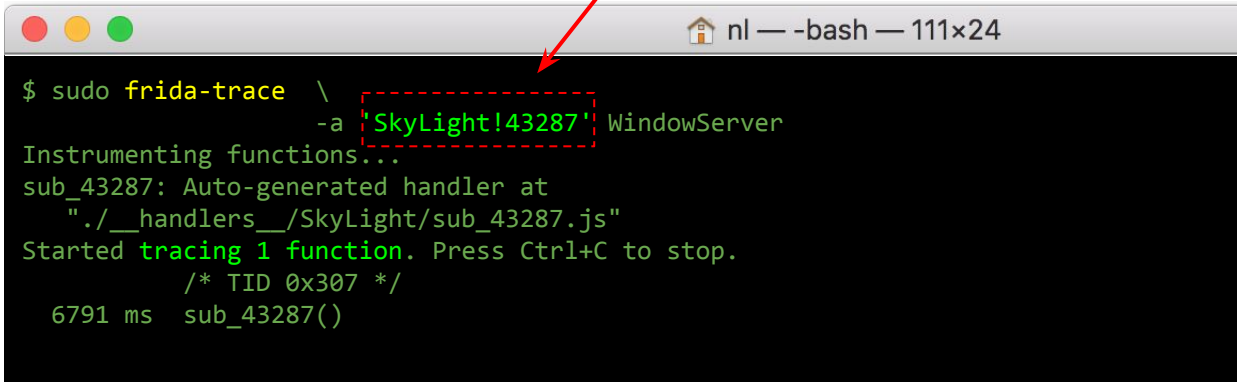
1. **Who created the file?**
2. **Why would malware set this attribute?**
3. **Why would malware create a file?**



On the endpoint

tracing functions

_XHWCaptureDesktop



1. Frida/DTrace lite-weight tracing frameworks
2. Proper debugging is too heavy
3. Thwarted by SIP on OS X (Good! But blunt)

On the endpoint

who did it?!

```
nl — -bash — 111×24

* thread #1, queue = 'com.apple.main-thread', stop reason = breakpoint 1.1
    frame #0: 0x00007fff544c3287 SkyLight`_XHWCaptureDesktop
SkyLight`_XHWCaptureDesktop:
->  0x7fff544c3287 <+0>: pushq  %rbp
    0x7fff544c3288 <+1>: movq   %rsp, %rbp
    0x7fff544c328b <+4>: pushq  %r15
    0x7fff544c328d <+6>: pushq  %r14
Target 0: (WindowServer) stopped.
(lldb) x/10wx $rdi
0x7ffee4c12610: 0x00001112 0x00000048 0x000153ab 0x0001249b
0x7ffee4c12620: 0x00000000 0x0000732a 0x00000000 0x00000001
0x7ffee4c12630: 0x00000000 0x00000000
```

1. Breakpoint in WindowServer
2. First parameter is the mach message
3. We can see the source port

4. Using source port we can get the PID/Task of the source process

```
nl — -bash — 111×24

$ sudo lsmp -a
Process (4460) : screencapture
  name      ipc-object    rights    identifier  type
---------   ----------  ----------  ----------- ------------
0x00000103  0x56e57599  send        0x00000000  TASK SELF (4460) screencapture
0x00000507  0x56e57a31  send        0x00000000  THREAD (0x7faf5)
0x00000603  0x56e56729  recv
                +       send-once   0x000153ab  (157) WindowServer
```

Can I defend myself?

on OS X

OS Mitigations

1) proposal: logging

Build a mechanism for allowing logging or trapping functions in SIP protected processes

Notification popup for any app that triggered screen capture

OS Mitigations

2) proposal: filter

Mach ports pass messages. It's high time that a firewall is developed to block certain messages on demand:

- Message ID 732Ah is a request for a screencapture.
- Could be defined in the same way as sandbox permissions:

```
(deny mach-msg
    (mach-msg-id 0x732a))
```

- Performance concerns balanced via ports/processes selection

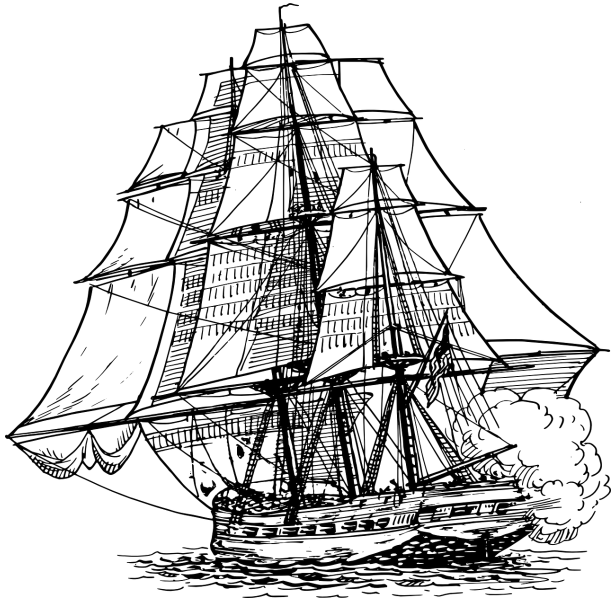
OS Mitigations

3) proposal: permissions

Only sandboxed applications with proper entitlements should be given access to API's that access pixels of other applications.*

* macOS Mojave: New data protections require apps to get user permission before using Mac camera and microphone (It's a start)

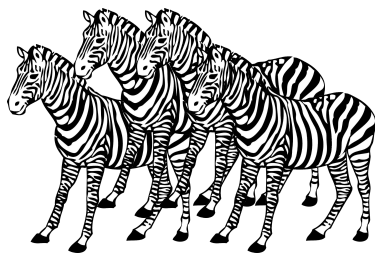
In closing



Some malware is thorny!

... but

> “In particular, our study concludes that **targeted malware does not use more** anti-debugging and anti-VM techniques than generic malware, although targeted malware tend to have a lower antivirus detection rate.”
>
> -- “Advanced or not?..,” P. Chen, et. al.



Conclusion

takeaways

- Lots of malware steals pixels
- Can do it direct through Mach Interfaces
- Sandboxes don’t help... Maybe a little.
- Proposed mitigations. Thoughts?

Questions and Answers

Thank you! :-)

Mikhail Sosonkin

http://debugtrap.com

@hexlogic

Thanks all!

- https://pixabay.com/
- http://highlight.hohli.com/
- **Objective-See**
- **Objective By The Sea**