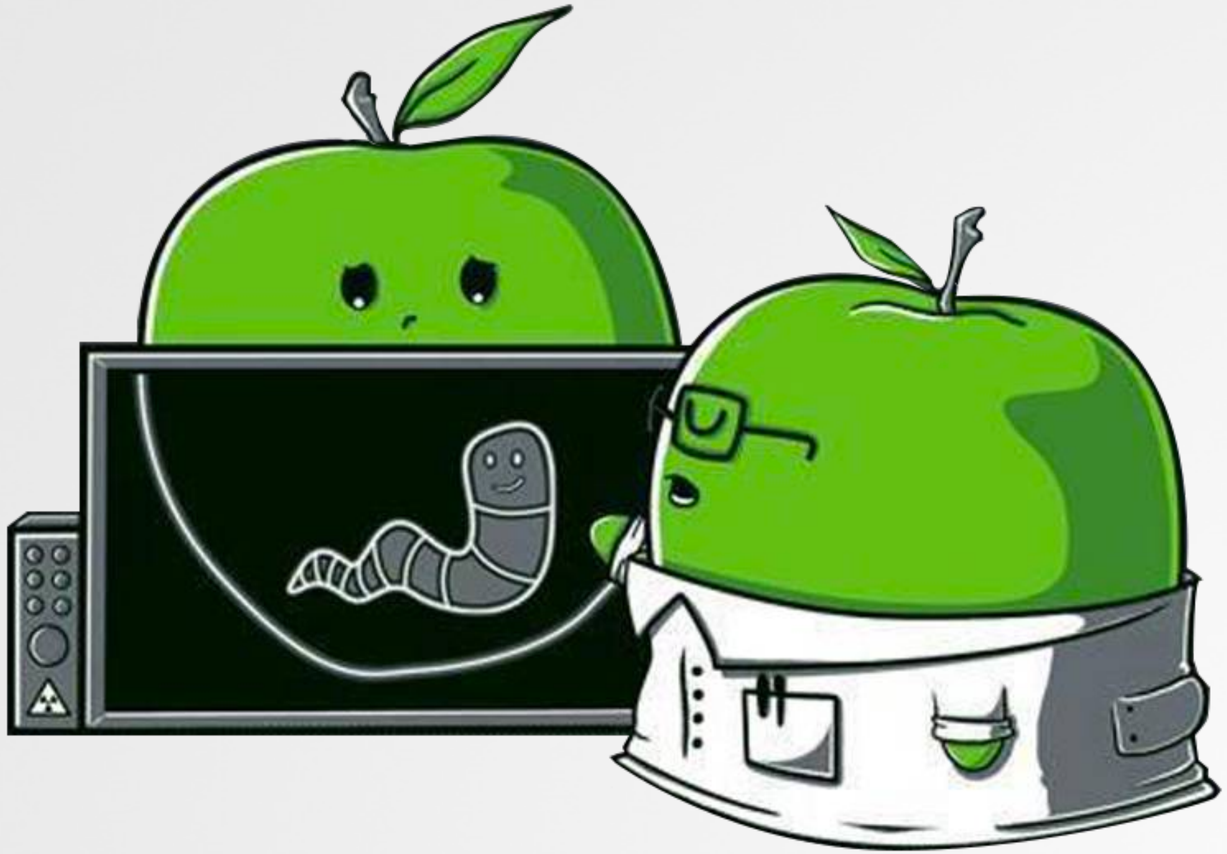
Déjà Vu

Uncovering Stolen Algorithms in Commercial Products



WHOIS



Patrick Wardle
(Objective-See Foundation)



Tom McGuire
(Johns Hopkins University)

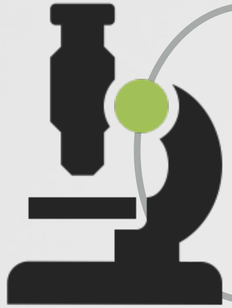
Outline



The Victim

The Hunt





Proving Equivalence



Resolutions

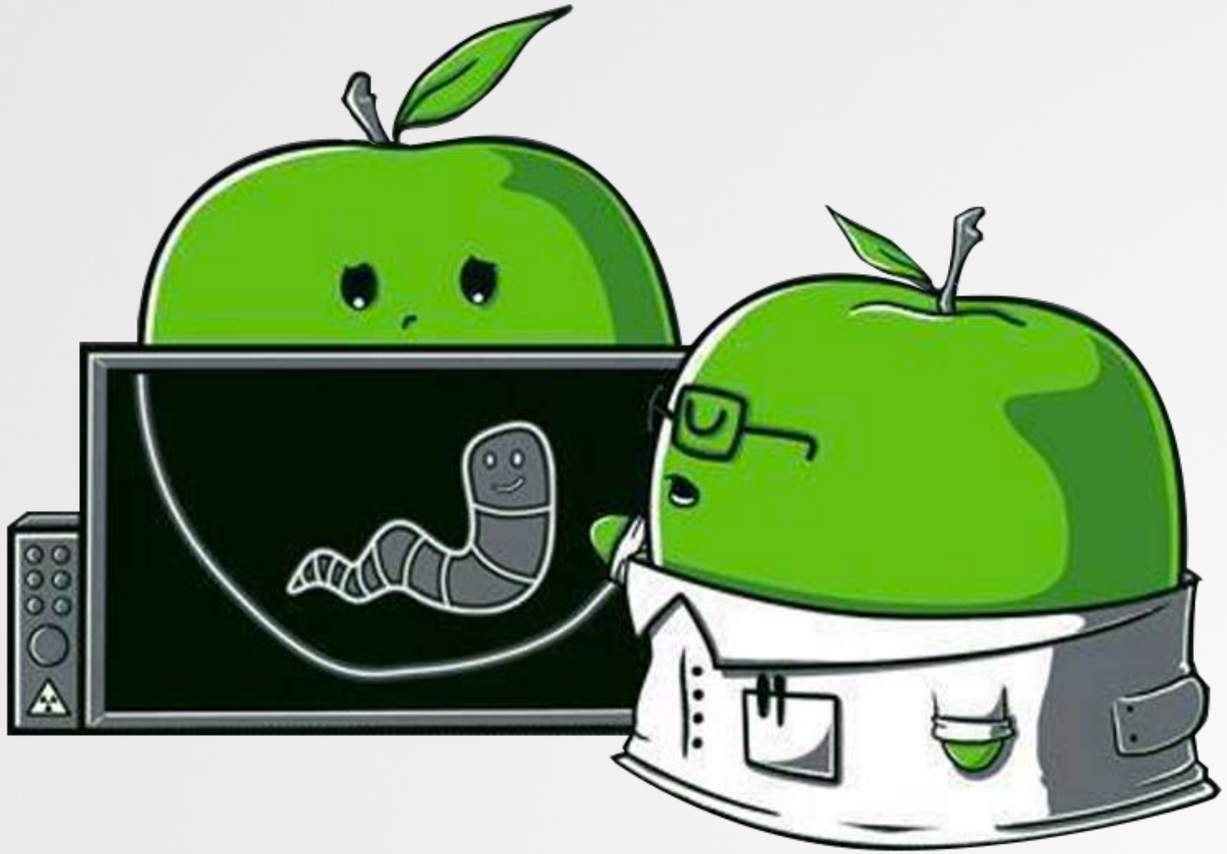
Goals:



Describe approaches to uncovering, confirming, and resolving, the unauthorized use of "stolen" code in commercial products.

The Victim

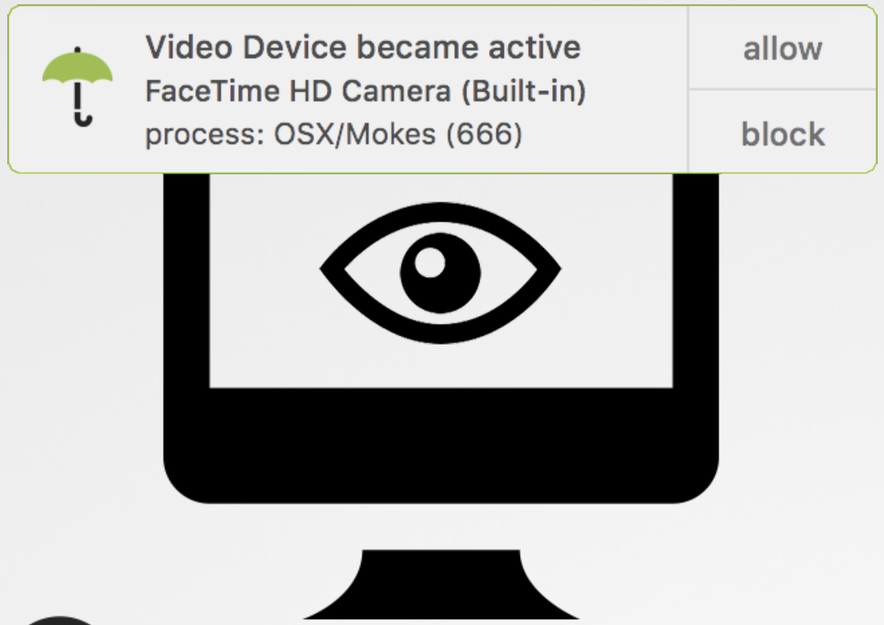
OverSight



OverSight

a mic & webcam monitor (macOS)





Mic & webcam monitor

"killer" feature

Identify process

Detect piggy-backing

Released @ Virus Bulletin 2016

--► free, but close-sourced
(open-sourced as of 2021)

...prior to any macOS protections :)

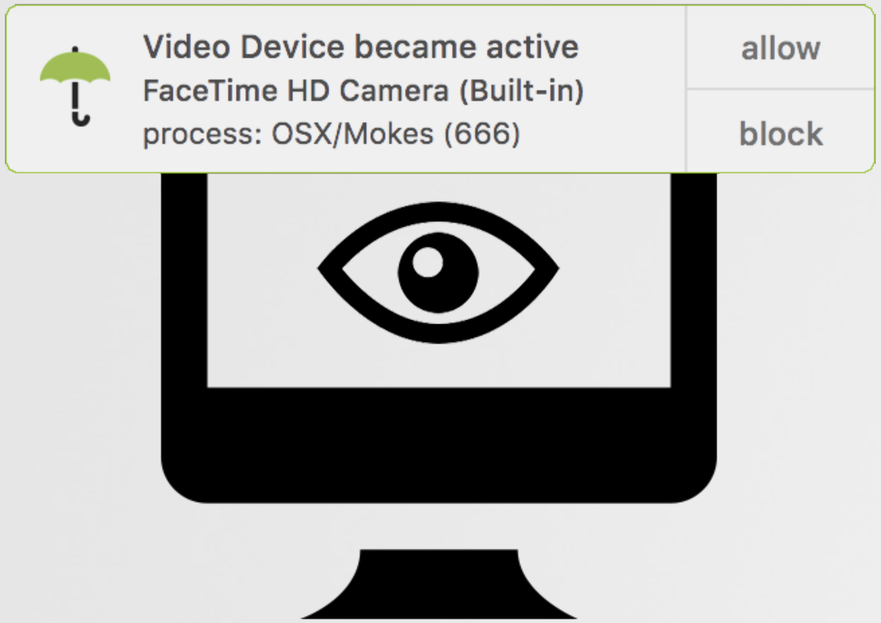
OverSight

vs. malware

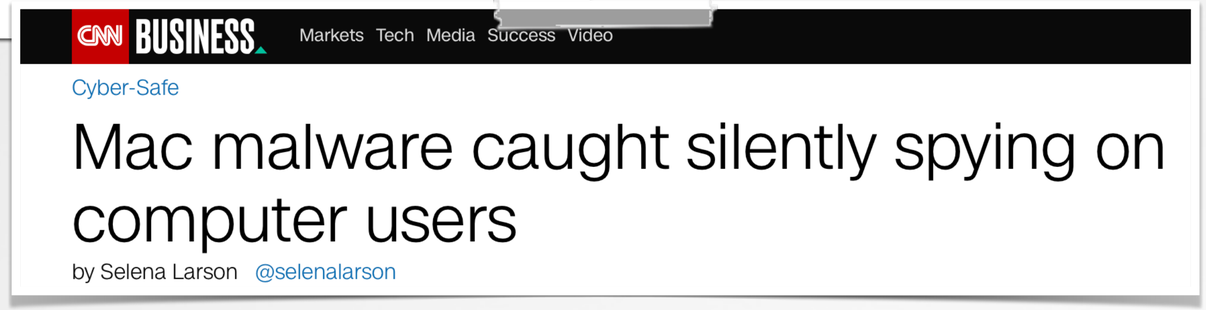
malware: Mokes

```
01   AVFMediaRecorderControl::setupSessionForCapture(void) proc
02    ...
03    call    AVFCameraSession::state(void)
04    call    AVFAudioInputSelectorControl::createCaptureDevice(void)
05    lea     rdx, "Could not connect the video recorder"
06    ...
07    call    QMediaRecorderControl::error(int,QString const&)
```

Webcam capture via AVFoundation



OverSight vs. Mokes



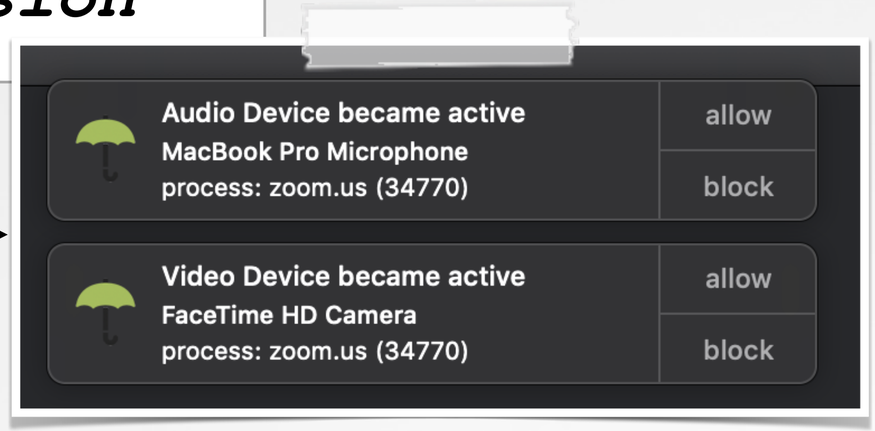
Webcam "aware" (macOS) malware:

- **Mokes**
- **Crisis**
- **Eleanor**
- **FruitFly**
- **... and many more!**

OverSight
vs. 0days vulnerabilities

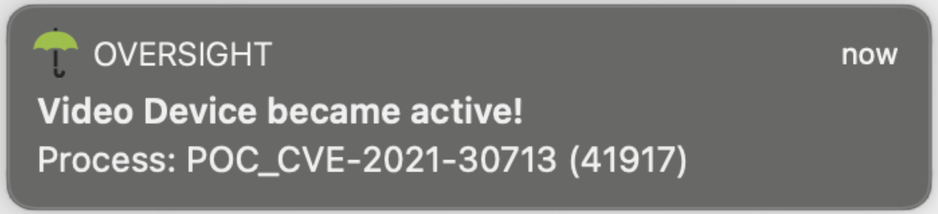
bug credit: Jonathan Leitschuh

"A [0day] vulnerability in the Mac Zoom Client allows any malicious website to enable your camera without your permission"



OverSight vs. 0day

OverSight vs. 0day



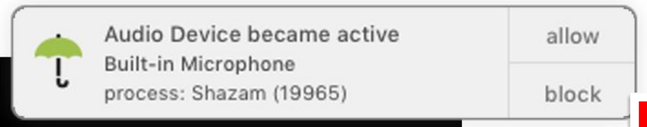
analysis: Jamf

"A zero-day [exploited by XCSSET malware] allows an attacker to bypass Apple's TCC protections which safeguard privacy"

OverSight
vs. legitimate apps

Toggling the app to the 'OFF' state, and manually invoking the `'isRecording'` method directly from the debugger, still returns YES:

```
(lldb) p (BOOL)[0x100729040 isRecording]
(BOOL) $19 = YES
```



Shazam, was always listening :/

SIGN IN The Register

{* SECURITY *}

Shhh! Shazam is always listening – even when it's been switched 'off'

Shazam Support <shazam.support@shazam.com>
to patrick

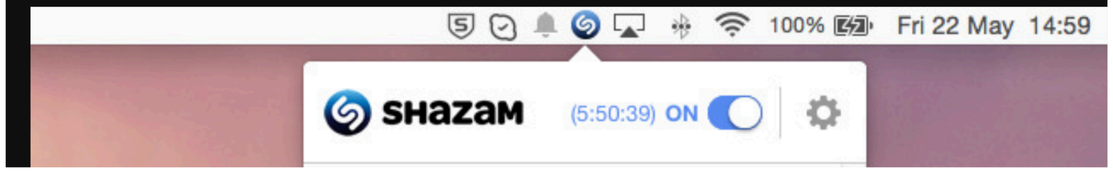
Hi Patrick,

Thanks for getting in touch and bringing this to our attention. The iOS and Mac apps use a shared SDK, hence the continued recording you are seeing on Mac.

We use this continued recording on iOS for performance, allowing us to deliver faster song matches to users.

it's a feature :\

Shazam promises Mac app update to end always-on listening



OverSight vs. Shazam



"Forget the NSA, it's Shazam that's always listening!"
objective-see.org/blog/blog_0x13.html

Process Identification

a must! but the APIs aren't helpful

```
01    CMIODeviceID cameraID = <id of built in camera>;
02    UInt32 isRunning = 0;
03    UInt32 propertySize = 0;
04    CMIOObjectPropertyAddress propertyStruct = {0};
05
06    propertySize = sizeof(isRunning);
07    propertyStruct.mScope = kCMIOObjectPropertyScopeGlobal;
08    propertyStruct.mSelector = kAudioDevicePropertyDeviceIsRunningSomewhere;
09
10    CMIOObjectGetPropertyData(cameraID, &propertyStruct, 0, NULL,
11        sizeof(kAudioDevicePropertyDeviceIsRunningSomewhere), &propertySize, &isRunning);
```

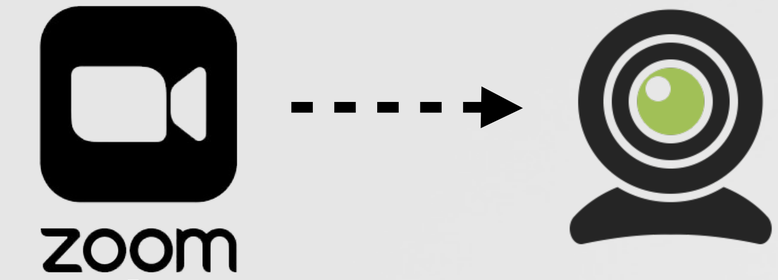
code for:
"is camera on"

The system APIs do not tell us:
which process is accessing the webcam (or microphone)



OverSight's Process Identification

step one: enumerate mach messages (ports)

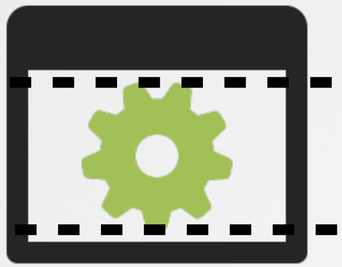



```
# lsmp -p <camera assistant>
...
0x00009f1f  0xbf7f289d send-once  ->
     0x0000000000000000 0x0001f60b  (34770) zoom.us
```

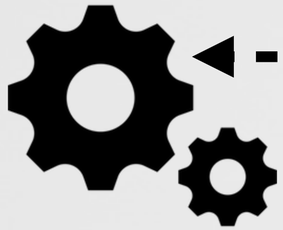
Listing mach ports
(via lsmp)











Camera/mic daemon
(eg. VDCAssistant,
AppleCameraAssistant, etc.)



/usr/bin/lsmp entitled:
"com.apple.system-task-ports.read"
(allows the listing of mach ports)

OverSight's Process Identification

programmatically enumerating "mach senders"

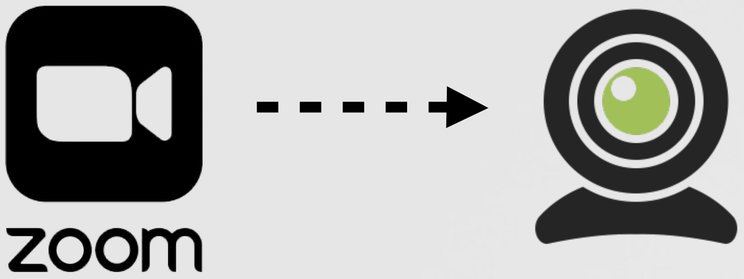
```
01  -(NSMutableDictionary*)enumMachSenders:(pid_t)targetProcess {
02
03  //exec 'lsmp' w/ pid of camera assistant to get mach ports
04  results = [[NSString alloc] initWithData:execTask(LSMP,
05                              @[@"-p", @(targetProcess).stringValue], YES) encoding:NSUTF8StringEncoding];
06  //parse results
07  // split on new line (then look for (<pid>) process name)
08  for(NSString* line in [results componentsSeparatedByCharactersInSet:
09                                 [NSCharacterSet characterSetWithCharactersInString:@"\n"]]) {
10
11      //skip blank lines
12      if(0 == line.length) continue;
13
14      //parse on '()'
15      subStrings = [line componentsSeparatedByCharactersInSet:
16                          [NSCharacterSet characterSetWithCharactersInString:@"()"]];
17
18      //sanity check
19      if(subStrings.count < 3) continue;
20
21      //extract process id (insides '()', so will be second substring)
22      processID = @([[subStrings objectAtIndex:0x1] integerValue]);
23
24      //add/inc to dictionary
25      senders[processID] = @([senders[processID] unsignedIntegerValue] + 1);
```

parse output

Executing lsmp (-p pid) & parsing its output

OverSight's Process Identification

step 2: query I/O registry (IOUserClientCreators)




```
# ioreg -l
...
| | +-o RootDomainUserClient
        <class RootDomainUserClient, id 0x10000099a ... >
    | | |    {
    | | |      "IOUserClientCreator" = "pid 34770, zoom.us"
    | | |    }
```

Listing "user clients"
(via ioreg)



I/O registry



Candidate processes

OverSight's Process Identification

programmatically querying the I/O registry

```
01  -(NSMutableDictionary*)enumDomainUserClients {
02
03   //get IOPMrootDomain obj
04   matchingService = IOServiceGetMatchingService(kIOMasterPortDefault, IOServiceMatching("IOPMrootDomain"));
05   ...
06
07   //get iterator
08   IORegistryEntryGetChildIterator(matchingService, kIOServicePlane, &iterator));
09
10   //iterator over all children (looking for 'IOUserClientCreator')
11   while((child = IOIteratorNext(iterator))) {
12
13    //get 'IOUserClientCreator'
14    creator = IORegistryEntryCreateCFProperty(child, CFSTR("IOUserClientCreator"), kCFAllocatorDefault, 0);
15
16    //parse
17    components = [creator componentsSeparatedByCharactersInSet:
18                          [NSCharacterSet characterSetWithCharactersInString:@" ,"]];
19
20    //extract pid and save
21    processID = [NSNumber numberWithShort:[components[0x1] intValue]];
22
23    //add/inc to dictionary
24    clients[processID] = @([clients[processID] unsignedIntegerValue] + 1);
```

parse results

Querying the I/O registry
("IOPMrootDomain/IOService/IOUserClientCreator")

OverSight's Process Identification

step 3: sample process and examine stack trace

```
01  #define SAMPLE @"/usr/bin/sample"
02
03  //invoke 'sample' to confirm candidates are using CMIO/video/av inputs
04  // note: audio/video invoke 'CMIOGraph::DoWork' (will be in call stack)
05  -(NSMutableArray*)sampleCandidates:(NSArray*)currentSenders {
06
07     ...
08
09     //exec 'sample' to get stack trace of process
10     results = [[NSString alloc] initWithData:execTask(SAMPLE,
11                     @[processID.stringValue, @"1"], YES) ...];
12
13     //check for 'CMIOGraph::DoWork'
14     // note: both audio/video invoke this
15     if(YES != [results containsString:@"CMIOGraph::DoWork"]) {
16
17        //save process
18     }
```

mic/webcam API's in stack trace?



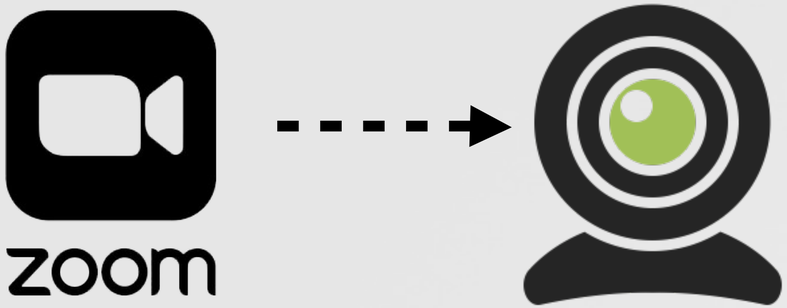
CMIOGraph::DoWork

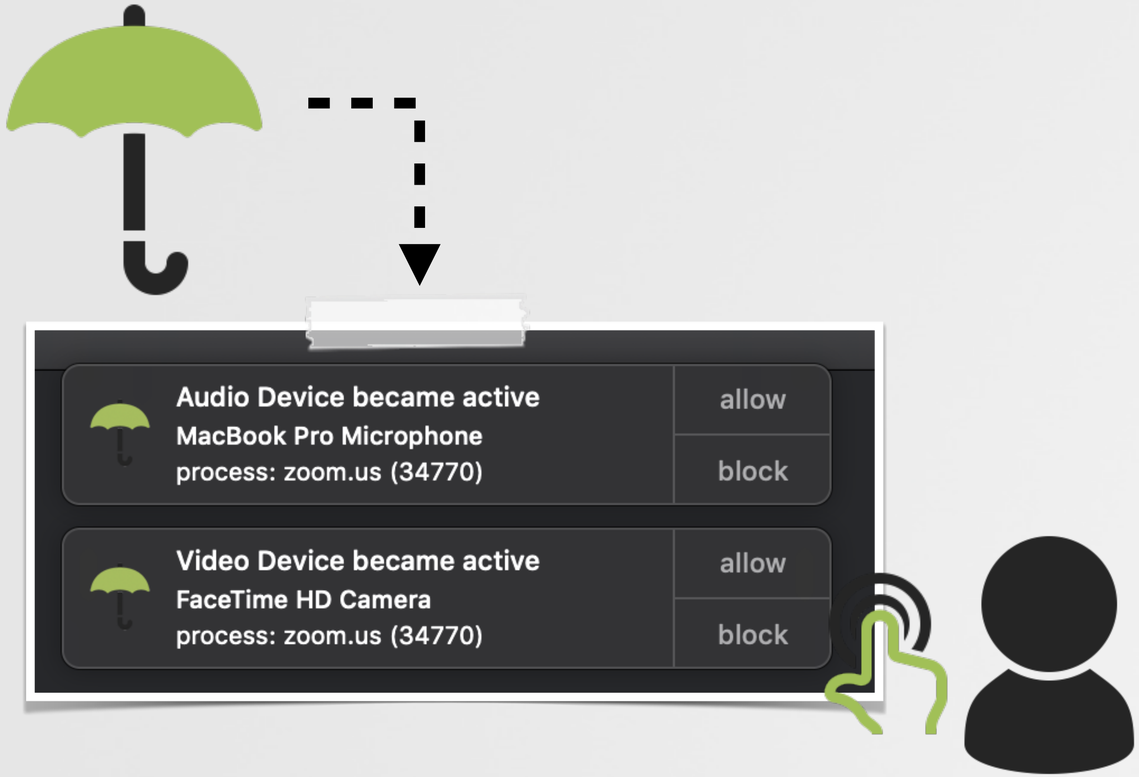
Executing sample (w/ pid) to obtain stack trace

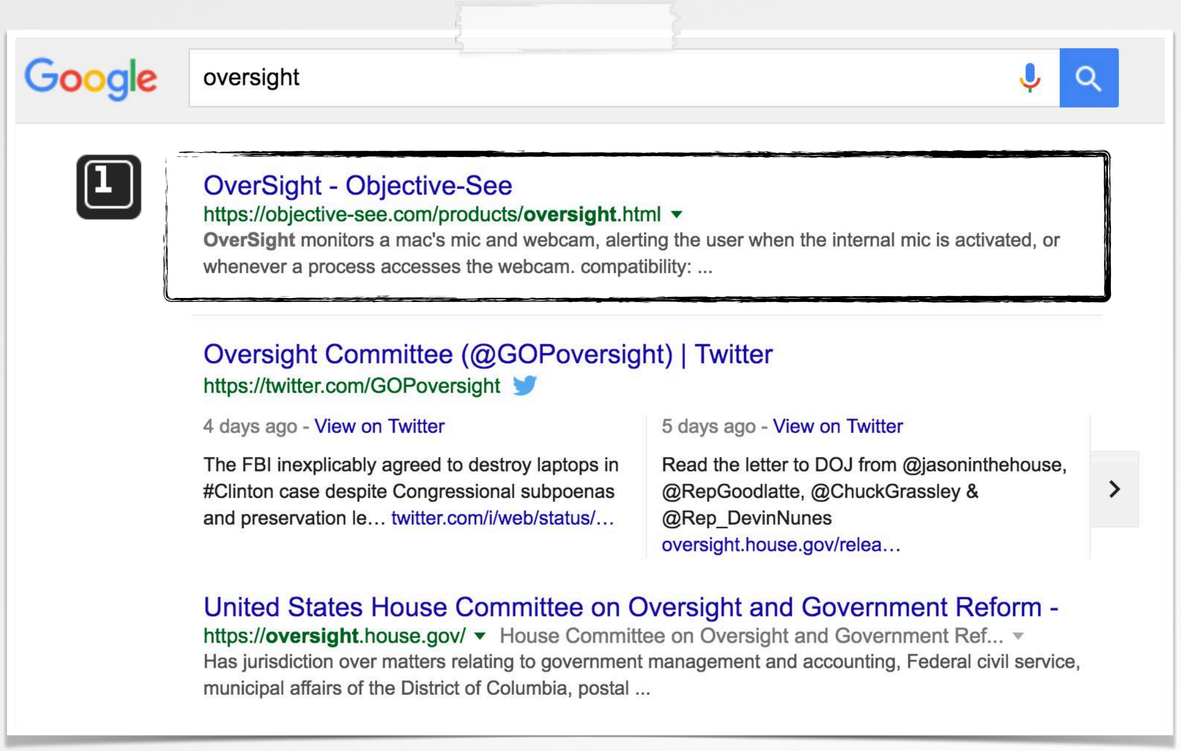
/usr/bin/sample: entitled with "task_for_pid-allow"
(which allows it to read (sample) other processes' memory)

Reliable Process Identification

...for both the mic & webcam



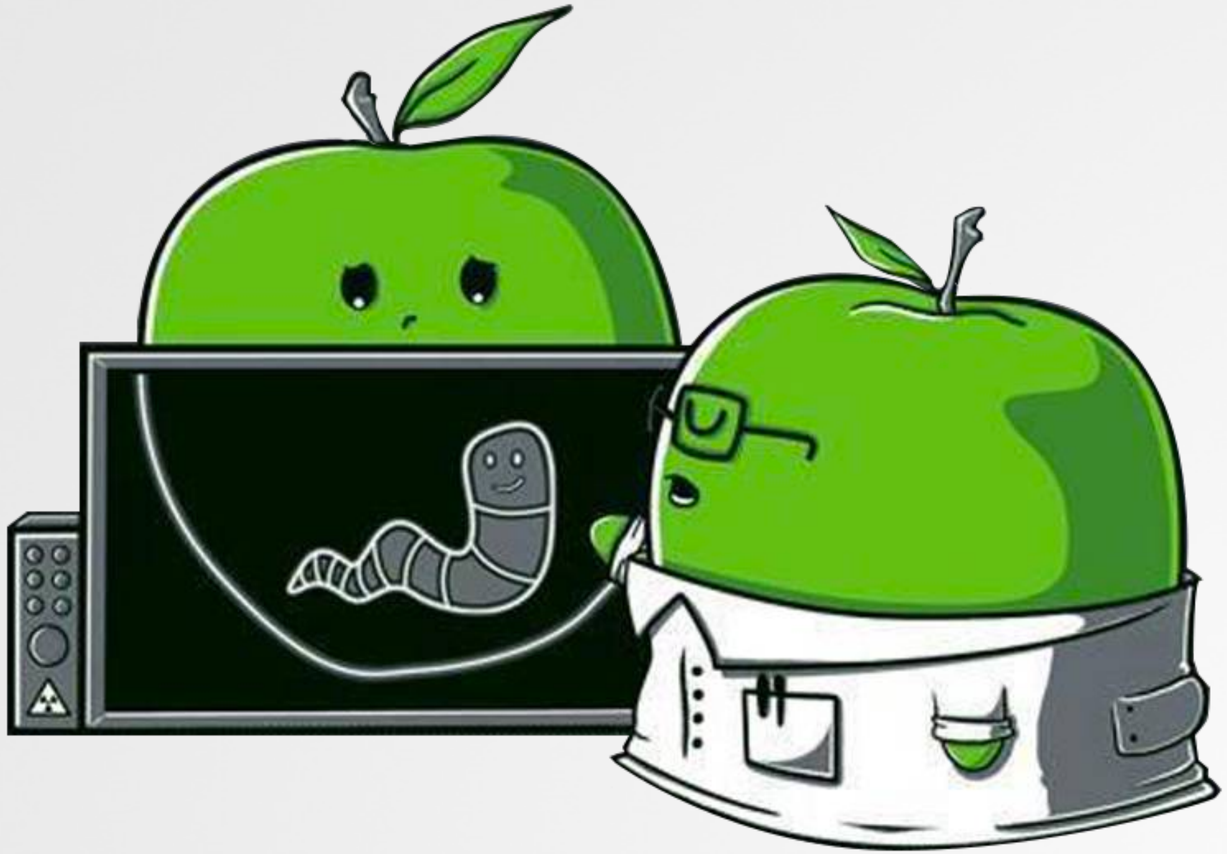




...and popularity!

The Hunt

OverSight ...in other products !?



How Would This Even Be Possible?

...by reverse-engineering the OverSight application



```objc
-(NSMutableDictionary*)enumMachSenders:(pid_t)targetProcess {
    ...
    NSArray* arguments = @[@"-p", @(targetProcess).stringValue];
    NSMutableDictionary* results = [NSMutableDictionary dictionary];

    NSData* results = execTask(@"/usr/bin/lsmp", arguments, YES);
```

Reconstructed code

Technically trivial,
...ethically & legally; questionable!

```objc
/* @class Enumerator */
-(void *)enumMachSenders:(int)arg2 {
    var_F0 = objc_autoreleasePoolPush();
    var_F8 = [[NSMutableDictionary dictionary] retain];
    r15 = [NSString alloc];
    var_40 = @"-p";
    var_E4 = arg2;
    rax = [NSNumber numberWithInt:arg2];
    rax = [rax retain];
    var_C8 = rax;
    r14 = [[rax stringValue] retain];
    *(&var_40 + 0x8) = r14;
    rax = [NSArray arrayWithObjects:&var_40 count:0x2];
    r12 = [rax retain];
    rbx = [_execTask(@"/usr/bin/lsmp", r12, 0x1) retain];
```

OverSight's disassembly

Article 6 of the 1991 EU Computer Programs Directive allows reverse engineering for the purposes of interoperability, but prohibits it for the purposes of creating a competing product, and also prohibits the public release of information obtained through reverse engineering of software



Free!
(OverSight)

Commercial
(...not OverSight)

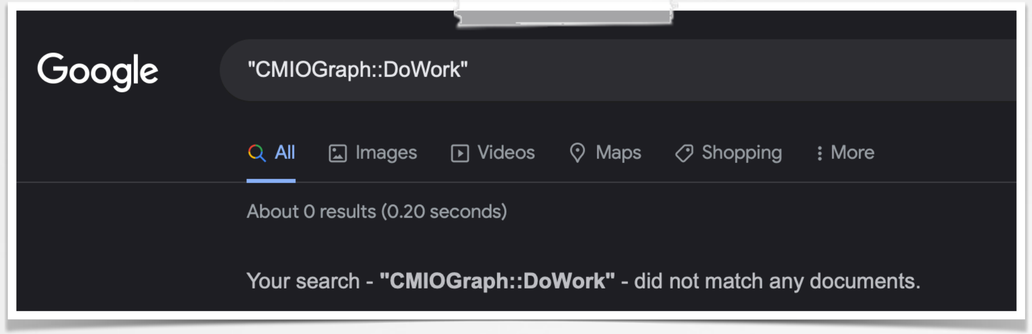
OverSight's Algorithm is Quite unique

...and is a bit janky (+ as we'll see, broke)

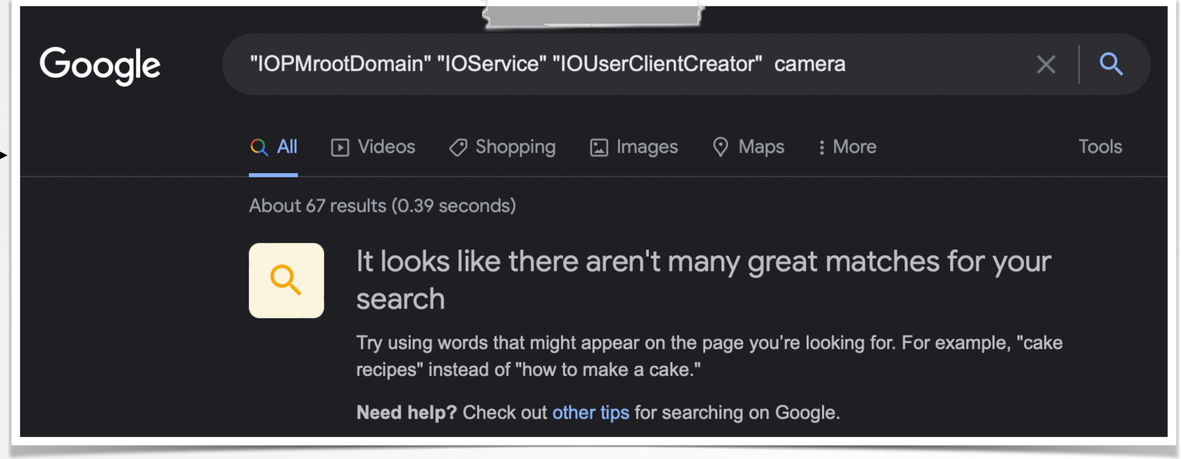
```
//parse results
// ->looking for (<pid>) process name
for(NSString* line in [results componentsSeparatedByCharactersInSet:[NSCharacterSet
    characterSetWithCharactersInString:@"\n"]])
{
    //skip blank lines
    if(0 == line.length)
    {
        //skip
        continue;
    }

    //parse on '()'
    subStrings = [line componentsSeparatedByCharactersInSet:[NSCharacterSet
        characterSetWithCharactersInString:@"()"]];
    if(subStrings.count < 3)
    {
        //skip
        continue;
    }

    //skip 'unknown' processes
    // output looks like "(-) Unknown Process"
    if(YES == [[subStrings objectAtIndex:0x1] isEqualToString:@"-"])
    {
        //skip
        continue;
    }

    //extract process id
    // ->insides '()', so will be second substring
    processID = @([[subStrings objectAtIndex:0x1] integerValue]);
    if(nil == processID)
    {
        //skip
        continue;
    }
```

Approach is a bit "janky"
(what, no regex?)

Approach is very unique:



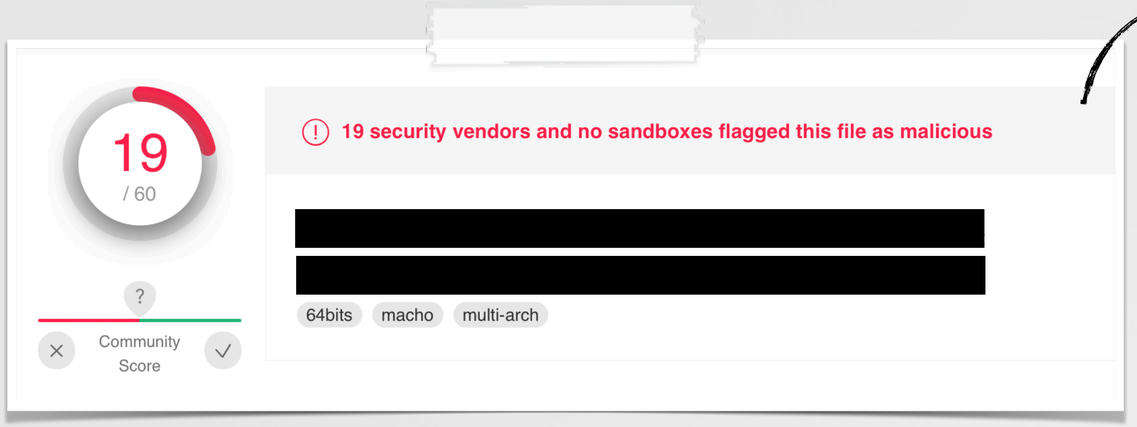
No Google results
(for "CMIOGraph::DoWork")



No "great matches"
(for I/O registry keys)

How It All Began (2018)

...via malware analysis



"Riskware" / "PUP"
(potential unwanted program)

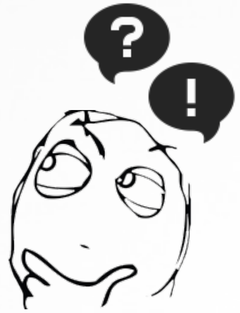


Flagged binary (on VirusTotal)

```
/* @class ZBAVMonitor */
-(void *)enumMachSendersForProcess:(void *)arg2 {
    r15 = arg2;
    var_110 = self;
    rax = *_NSFoundationVersionNumber;
    xmm0 = intrinsic_ucomisd(*double_value_1151_16, *rax);
    if (xmm0 > 0x0) {
            r13 = [[NSBundle mainBundle] pathForResource:@"lsmp" ofType:0x0];
    }
    else {
            r13 = @"/usr/bin/lsmp";
    }
```

method "**enumMachSendersForProcess:**"
which execs **/usr/bin/lsmp** !?

...just like OverSight !?



Familiar code?

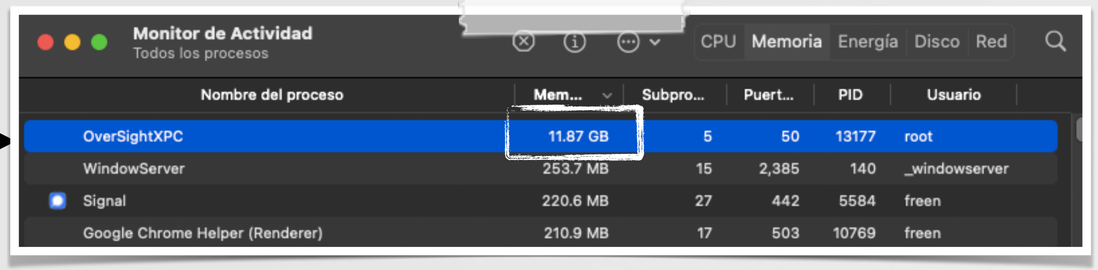
HUNTING VIA GOOGLE

bug(s) in OverSight ...in others too!?

OverSight using huge amount of memory #3

Open · Sr-Gonzalo opened this issue on Feb 20, 2021 · 14 comments

1



Camera always shows as active on M1 MacBook Pro #7

Open · mourke opened this issue on Apr 30, 2021 · 1 comment

2

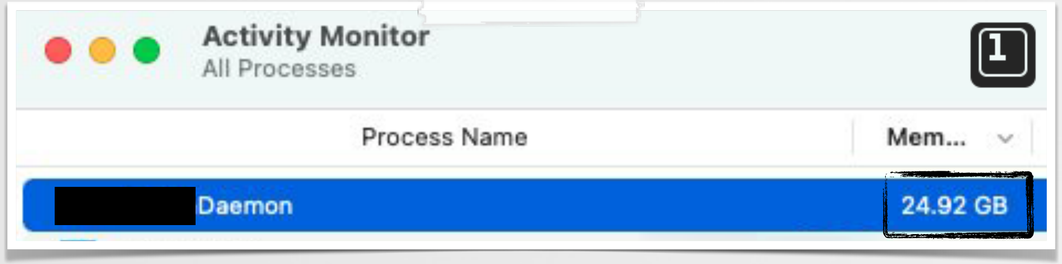


mourke commented on Apr 30, 2021 · edited

My camera is always shown as being on even when I freshly boot into macOS and nothing can be using it. I'm on a completely fresh install on my new MacBook. As a result notifications are not always sent when the camera actually becomes active.

Bugs in OverSight

Same bug(s)!?
...but these aren't OverSight!



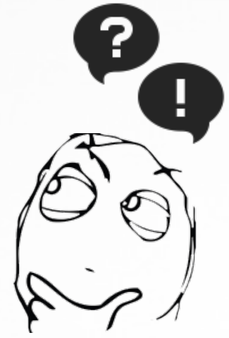
1

Various changes in macOS triggered issues in other (non-Objective-See) programs too !?

Known Issues

2

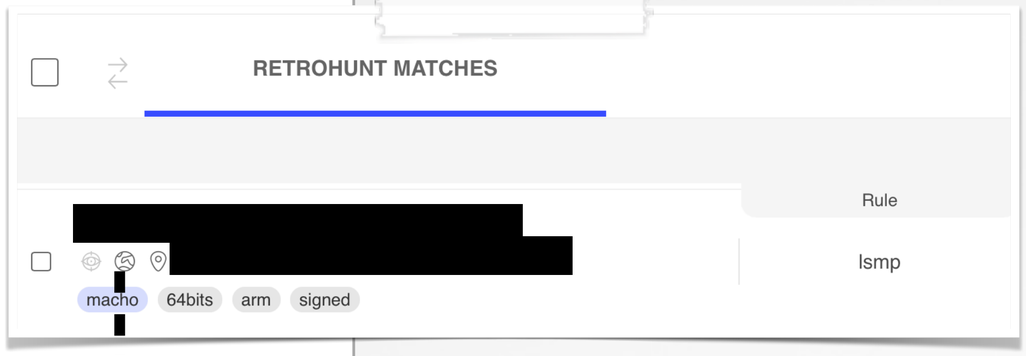
- Product will show that camera is "active" on recent versions of macOS, even though camera is not in use.



Hunting via Yara Rules

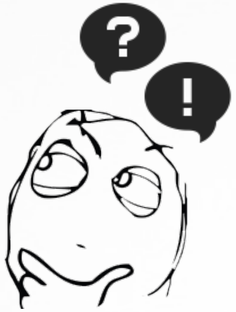
...and running across corpuses of binaries

```
private rule Macho {
    condition:
        uint32(0) == 0xfeedface or uint32(0) == 0xcefaedfe ...
}

rule lsmp {

   strings:
     $a = "lsmp"
   condition:
     Macho and $a
}
```



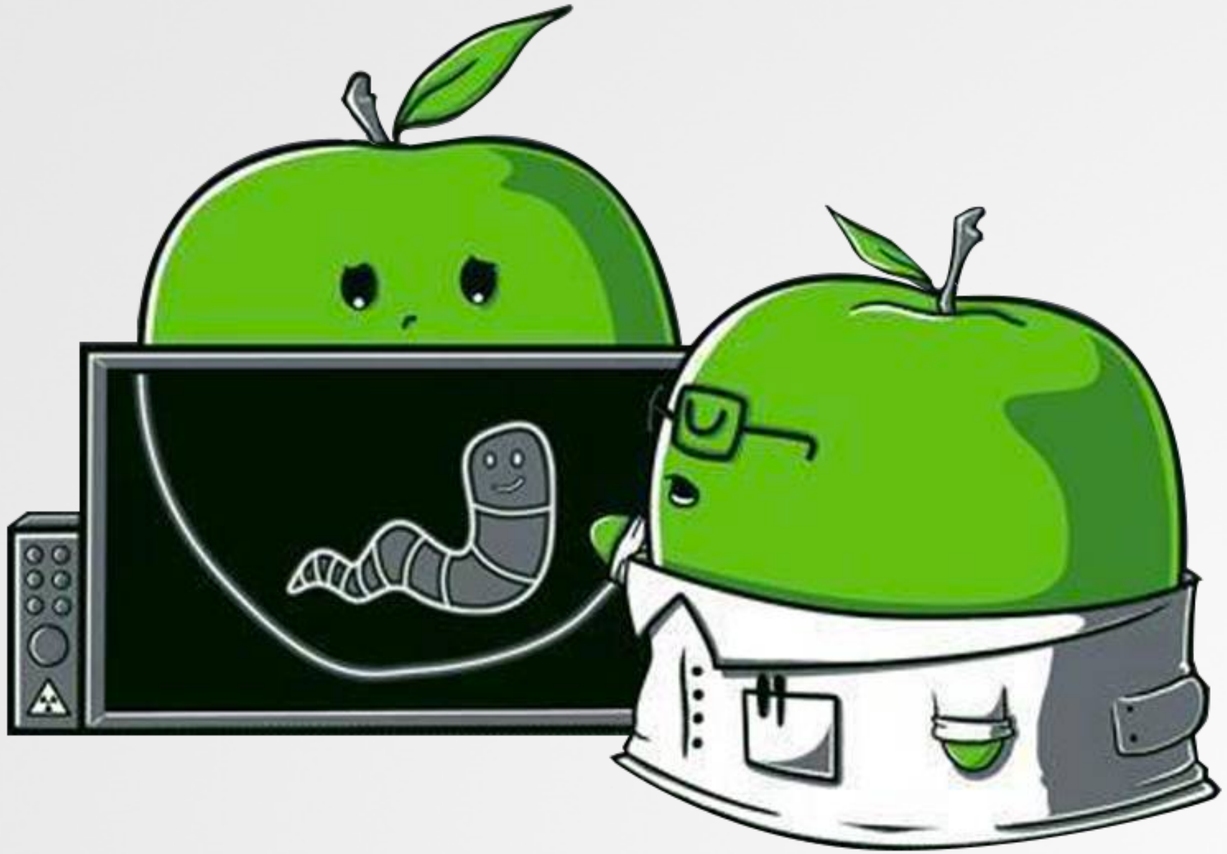

```
/* @class VedioProcessStat */
-(int)getVDCDictionaryWithPid:(int)arg2 {
    var_50 = r28;
    stack[-88] = r27;
    r31 = r31 + 0xfffffffffffffa0 - 0x7c0;
    r19 = [[NSString stringWithFormat:@"lsmp -p %d", r3] retain];
    var_798 = arg0;
    r0 = [arg0 excuteShellAndReturnOut:r19];
```

A match, with method named
"getVDCDictionaryWithPid:"
(that invokes lsmp)



Confirming Equivalency

...the code doesn't lie !



PRODUCT #1

use of lsmp & parsing

not OverSight

method: "enumMachSenders:"

```
/* @class Enumerator */
-(void *)enumMachSenders:(int)arg2 {
    var_F0 = objc_autoreleasePoolPush();
    var_F8 = [[NSMutableDictionary dictionary] retain];
    r15 = [NSString alloc];
    var_40 = @"-p";                                          [1]
    var_E4 = arg2;
    rax = [NSNumber numberWithInt:arg2];
    rax = [rax retain];
    var_C8 = rax;
    r14 = [[rax stringValue] retain];
    *(&var_40 + 0x8) = r14;
    rax = [NSArray arrayWithObjects:&var_40 count:0x2];
    r12 = [rax retain];
    rbx = [_execTask(@"/usr/bin/lsmp", r12, 0x1) retain];
```

```
loc_100005ff6:
    r14 = [(r14)(@class(NSCharacterSet),
                 @selector(characterSetWithCharactersInString:), @"()")]   [2]
    r15 = [[rbx componentsSeparatedByCharactersInSet:r14] retain];
    [var_C8 release];
    [r14 release];
    if ([r15 count] >= 0x3) goto loc_100006074;                            [3]
```

```
loc_100006074:
    r12 = @selector(objectAtIndex:);
    rbx = [_objc_msgSend_10000d0e8(r13, r12, 0x1) retain];
    r14 = [rbx isEqualToString:@"-"];                                      [4]
```

OverSight

method: "enumMachSendersForProcess:"

```
/* @class ZBAVMonitor */
-(void *)enumMachSendersForProcess:(void *)arg2 {
    r15 = arg2;
    var_110 = self;
    if (*double_value_1151_16 > **_NSFoundationVersionNumber) {
            r13 = [[NSBundle mainBundle] pathForResource:@"lsmp" ofType:0x0];
    }
    else {
            r13 = @"/usr/bin/lsmp";                                        [1]
    }

    r14 = [[NSTask alloc] init];
    r12 = [[NSPipe alloc] init];
    [r14 setLaunchPath:r13];
    var_40 = @"-p";
    *(&var_40 + 0x8) = [NSString stringWithFormat:@"%d", [r15 intValue]];
    [r14 setArguments:[NSArray arrayWithObjects:@"%d" count:0x2]];
    [r14 setStandardOutput:r12];

    [r14 launch];
```

```
var_138 = [NSCharacterSet characterSetWithCharactersInString:@"()"];   [2]
rbx = [*(var_178 + r15 * 0x8) componentsSeparatedByCharactersInSet:var_138];

if ([rbx count] >= 0x3 &&                                              [3]
    [[rbx objectAtIndex:0x1] isEqualToString:@"-"] == 0x0) {           [4]
```

Same:

1. Invoke lsmp (-p <pid>)
2. Split line "()"
3. Ignore if less than 3 substrings
4. Check for "-" character

Product #1

I/O registry query & parsing

not OverSight

method: "enumDomainUserClients:"

```
/* @class Enumerator */
-(void *)enumDomainUserClients {
    r14 = objc_autoreleasePoolPush();
    r15 = [[NSMutableDictionary dictionary] retain];
    rbx = *(int32_t *)*_kIOMasterPortDefault;
    rbx = IOServiceGetMatchingService(rbx, IOServiceMatching("IOPMrootDomain"));
    if (rbx != 0x0) {
        var_2C = 0x0;
        rax = IORegistryEntryGetChildIterator(rbx, "IOService", &var_2C);
        if (rax == 0x0) {
            var_50 = r15;
            var_B0 = r14;
            var_38 = 0x0;
            var_48 = 0x0;
            do {
                r15 = IOIteratorNext(0x0);
                if (r15 == 0x0) {
                    break;
                }
                r14 = IORegistryEntryCreateCFProperty(r15, @"IOUserClientCreator",
                                                      **_kCFAllocatorDefault, 0x0);
                IOObjectRelease(r15);
                if (r14 == 0x0) {
                    continue;
                }
                var_3C = rbx;
                r15 = [[NSCharacterSet characterSetWithCharactersInString:@" ,"] retain];
                rbx = [[r14 componentsSeparatedByCharactersInSet:r15] retain];
```

1

2

method: "enumDomainUserClients:"

```
/* @class ZBAudioCaptureMonitor */
-(void *)enumDomainUserClients {
    r15 = self;
    var_78 = [NSNumber numberWithInt:[[NSProcessInfo processInfo] processIdentifier]];
    rbx = [NSCountedSet set];
    r13 = *(int32_t *)*_kIOMasterPortDefault;
    rax = IOServiceMatching("IOPMrootDomain");
    rax = IOServiceGetMatchingService(r13, rax);
    if (rax != 0x0) {
        var_48 = rbx;
        var_60 = r15;
        var_2C = 0x0;
        rax = IORegistryEntryGetChildIterator(rax, "IOService", &var_2C);
        if (rax == 0x0) {
            var_40 = [NSMutableSet set];
            rbx = IOIteratorNext(var_2C);
            if (rbx != 0x0) {
                do {
                    r14 = IORegistryEntryCreateCFProperty(rbx, @"IOUserClientCreator",
                                                          **_kCFAllocatorDefault, 0x0);
                    IOObjectRelease(rbx);
                    if (r14 != 0x0) {
                        rbx = [r14 componentsSeparatedByCharactersInSet:
                                   [NSCharacterSet characterSetWithCharactersInString:@" ,", 0x0
```

1

2

Same:

1. **Query: "IOPMrootDomain/IOService/IOUserClientCreator"**
2. **Split on " ,"**

OverSight

Product #1

sample process & string matching

not OverSight



```
r14 = r15;
r15 = [_execTask(@"/usr/bin/sample", rax, 0x1) retain];                    1

(r15)(var_120, @selector(deleteSampleFile:), var_110, rcx, 0x10);          2

rdx = @"CMIOGraph::DoWork";
rax = (r15)(rbx, @selector(containsString:), rdx, rcx, 0x10);              3
var_C8 = rbx;
if (rax != 0x1) goto loc_100006a36;
```



```
rbx = [[NSTask alloc] init];
[rbx setLaunchPath:@"/usr/bin/sample"];                                    1

rax = [NSArray arrayWithObjects:&var_50 count:0x5];
[rbx setArguments:rax];
[rbx launch];
[rbx waitUntilExit];
[rbx release];

[[NSString stringWithContentsOfFile:r15 encoding:0x4 error:0x0]
                                    rangeOfString:@"CMIOGraph::DoWork"];   3
rbx = @"CMIOGraph::DoWork" != 0x0 ? 0x1 : 0x0;

[r14 closeFile];
[[NSFileManager defaultManager] removeItemAtPath:r15 error:0x0];           2
```

Same:

1. Sample (candidate) process
2. Delete sample's output file
3. Look for "CMIOGraph::DoWork"

OverSight

Product #2

use of lsmp & parsing

not OverSight

```
rbx = [[NSCharacterSet characterSetWithCharactersInString:@"\n"] retain]
r12 = [[r13 componentsSeparatedByCharactersInSet:rbx] retain];
```

1

```
rax = [r12 countByEnumeratingWithState:&var_1B0 objects:&var_C0 count:0x10];
if (rax == 0x0) goto loc_100006258;
```

2

```
rax = Foundation.CharacterSet.newlines.getter();
rbx = (extension in Foundation):Swift.StringProtocol.components
                    (separatedBy:)(r12, *type metadata for Swift.String, rax);
```

1

```
rax = *(rbx + 0x10);
if (rax == 0x0) goto loc_1000148f3;
```

2

```
r14 = [(r14)(@class(NSCharacterSet),
            @selector(characterSetWithCharactersInString:), @"()") retain]
r15 = [[rbx componentsSeparatedByCharactersInSet:r14] retain];
```

3

```
if ([r15 count] >= 0x3) goto loc_100006074;
```

4

```
rax = Foundation.CharacterSet.init('()', 0xe200000000000000);

rdx = var_90;
rax = Swift.StringProtocol.components(separatedBy:)(var_68, r13, rdx);
```

3

```
if (*(rax + 0x10) < 0x3) goto loc_100013c70;
```

4

Same:

1. Split (lsmp) output on newlines ("\n")
2. Leave if no lines (count = 0)
3. Split line "()"
4. Ignore if less than 3 substrings

OverSight

Product #2

I/O registry query & parsing

not OverSight

```
rbx = IOServiceGetMatchingService(rbx, IOServiceMatching("IOPMrootDomain"));   1
var_2C = 0x0;
rax = IORegistryEntryGetChildIterator(rbx, "IOService", &var_2C);   2
do {
    r15 = IOIteratorNext(0x0);
    if (r15 == 0x0) {
        break;
    }
    r14 = IORegistryEntryCreateCFProperty(r15, @"IOUserClientCreator",   3
                                          **_kCFAllocatorDefault, 0x0);
    var_3C = rbx;
    r15 = [[NSCharacterSet characterSetWithCharactersInString:@" ,"] retain];   4
    rbx = [[r14 componentsSeparatedByCharactersInSet:r15] retain];
```

```
rax = Swift.String.utf8CString.getter('IOPMroot', 'Domain\x00\xee');   1
r14 = IOServiceMatching(rax + 0x20);
r12 = *(int32_t *)*_kIOMasterPortDefault;
rax = [r14 retain];
r14 = rax;
rax = IOServiceGetMatchingServices(r12, rax, &var_30);
```

```
rax = (extension in Foundation):Swift.String._bridgeToObjectiveC()('IOService');   2
r15 = objc_retainAutorelease(rax);
r14 = [rax UTF8String];
```

"IOUserClientCreator"

```
rax = Swift.String._bridgeToObjectiveC()(0xd000000000000013, 0x8000000010006e8a0)   3
r15 = IORegistryEntryCreateCFProperty(r14, rax, r12, 0x0);
[rax release];
```

```
Foundation.CharacterSet.init(',', 0xe100000000000000);   4
r15 = Swift.StringProtocol.components(separatedBy:)(r12, r15, rax);
```

Same:

1. Matching Service: "IOPMrootDomain"
2. Get Child Iterator: "IOService"
3. Get Property Value: "IOUserClientCreator"
4. Parse result, by splitting on " ,"

OverSight

Product #3

use of lsmp & parsing

not OverSight

```
r15 = [NSString alloc];
var_40 = @"-p";
var_E4 = arg2;
rax = [NSNumber numberWithInt:arg2];
rax = [rax retain];
var_C8 = rax;
r14 = [[rax stringValue] retain];
*(&var_40 + 0x8) = r14;
rax = [NSArray arrayWithObjects:&var_40 count:0x2];
r12 = [rax retain];
rbx = [_execTask(@"/usr/bin/lsmp", r12, 0x1) retain];
```

1

```
rbx = [[NSCharacterSet characterSetWithCharactersInString:@"\n"] retain];
r12 = [[r13 componentsSeparatedByCharactersInSet:rbx] retain];
```

2

```
/* @class VedioProcessStat */
-(void *)getVDCDictionaryWithPid:(int)arg2 {
    rcx = arg2;
    r15 = *_objc_msgSend;
    rax = [NSString stringWithFormat:@"lsmp -p %d", rcx];
    rax = [rax retain];
    var_638 = self;
    r14 = r15;
    var_6C0 = [[self excuteShellAndReturnOut:rax] retain];
    rbx = [(r14)(var_6C0, @selector(componentsSeparatedByString:),
                                                @"\n", rcx) retain];
```

1

2

Same:

1. Invoke lsmp (-p <pid>)
2. Split on new lines ("\n")

OverSight

Product #3

I/O registry query & parsing

not OverSight

```
rbx = IOServiceGetMatchingService(rbx, IOServiceMatching("IOPMrootDomain")); 1
var_2C = 0x0;
rax = IORegistryEntryGetChildIterator(rbx, "IOService", &var_2C); 2
do {
    r15 = IOIteratorNext(0x0);
    if (r15 == 0x0) {
        break;
    }
    r14 = IORegistryEntryCreateCFProperty(r15, @"IOUserClientCreator",
                                          **_kCFAllocatorDefault, 0x0); 3
```

```
rax = IOServiceMatching("IOPMrootDomain"); 1
rax = IOServiceGetMatchingService(*(int32_t *)*_kIOMasterPortDefault, rax);
var_144 = 0x0;
rax = IORegistryEntryGetChildIterator(rax, "IOService", &var_144); 2
if (rax == 0x0) {
        rax = IOIteratorNext(var_144);
        if (rax != 0x0) {
            do {
                r14 = IORegistryEntryCreateCFProperty(rbx, @"IOUserClientCreator",
                                                      **_kCFAllocatorDefault, 0x0); 3
```

Same:

1. Matching Service: "IOPMrootDomain"
2. Get Child Iterator: "IOService"
3. Get Property Value: "IOUserClientCreator"

OverSight

PRODUCT #3

sample process & string matching

not OverSight

```
r14 = r15;
r15 = [_execTask(@"/usr/bin/sample", rax, 0x1) retain];

(r15)(var_120, @selector(deleteSampleFile:), var_110, rcx, 0x10);

rdx = @"CMIOGraph::DoWork";
rax = (r15)(rbx, @selector(containsString:), rdx, rcx, 0x10);
var_C8 = rbx;
if (rax != 0x1) goto loc_100006a36;
```

1

2

```
int ___31-[OwlManageDaemon sampleVedio:]_block_invoke(int arg0) {
    r15 = arg0;
    r14 = *(arg0 + 0x20);
    rcx = *(int32_t *)(r15 + 0x38);
    rax = [NSString stringWithFormat:@"sample %d 0.7 50", rcx];
    rax = [rax retain];
    r14 = [[r14 excuteShellAndReturnOut:rax] retain];
    [rax release];
    if ([r14 containsString:@"CMIOGraph::DoWork"] != 0x0) {
            rax = @(0x1);
    }
```

1

2

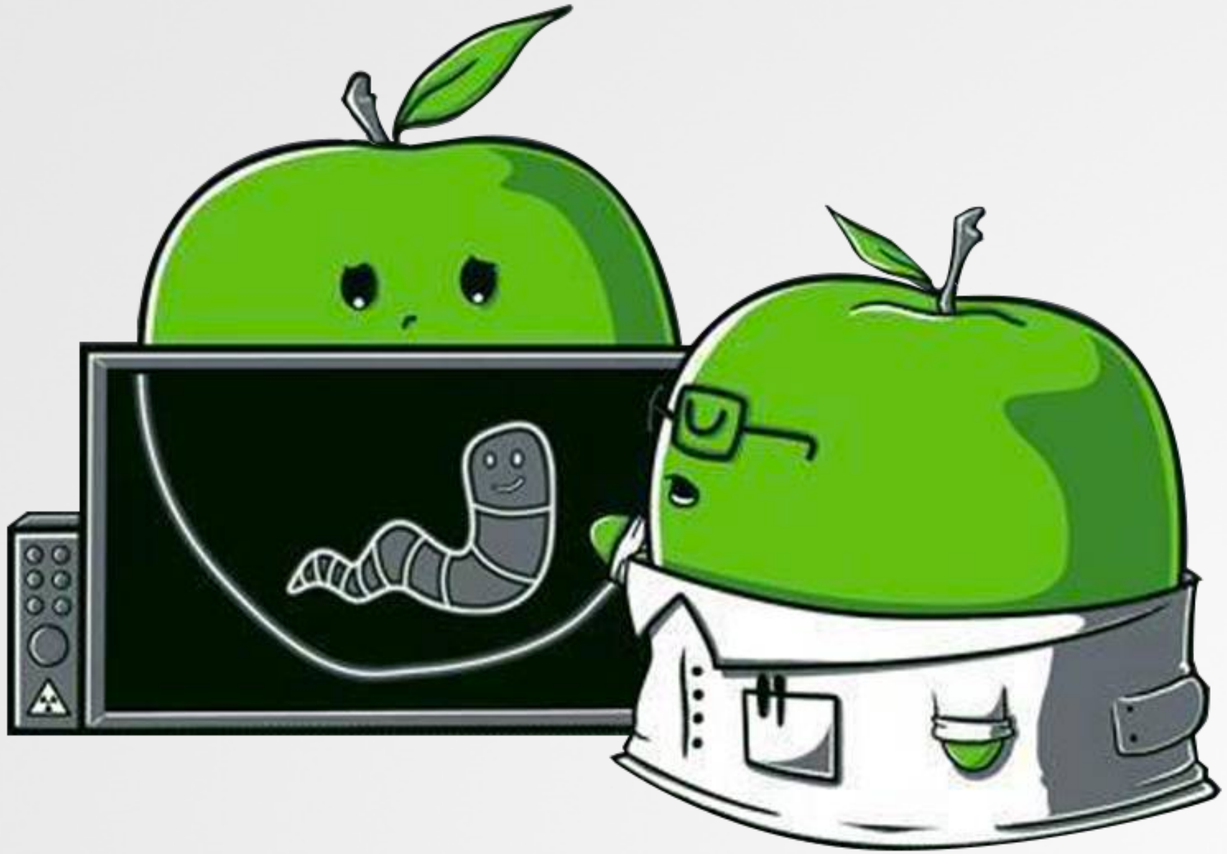
Same:

1. Invoke sample
2. Look for "CMIOGraph::DoWork" (in output from sample)

OverSight

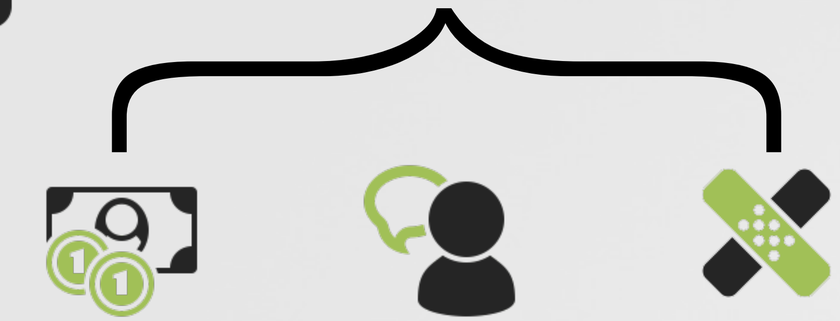
Resolutions

and now what?

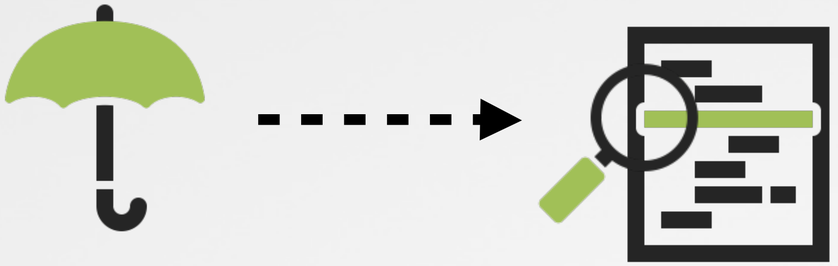


Approach
to resolve the matter

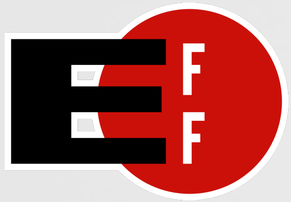
1 **Define your goals**



2 **Create Proof**



3 **Consult a lawyer** → optional, but recommended !



eff.org

4 **Reach out directly**



→ **Professionally**

+ your proof

+ goals for resolution

Reasonable Resolutions

...what corporations (generally) want



An amicable resolution



The majority of cases, the infringement is the work of a single (naive?) developer vs. malice of the entire corporation.



A license agreement → covering them legally



Non-disparagement → "optics"

...and will pay!



Win-Win Resolution

ack'd + code removal & financial compensation

Aloha Patrick,

Firstly, I'm truly appreciate you for feedback and this finding. As we are strongly commited with copyrights and code ownership, it was sad to know that we have a part of propriety logic in our product. So we made internal investigation and have determined some actions to do to prevent such situations in the future.

1 Acknowledgement

2 Code Removal

Also we will remove this logic from [redacted] and update existed users to this new build.

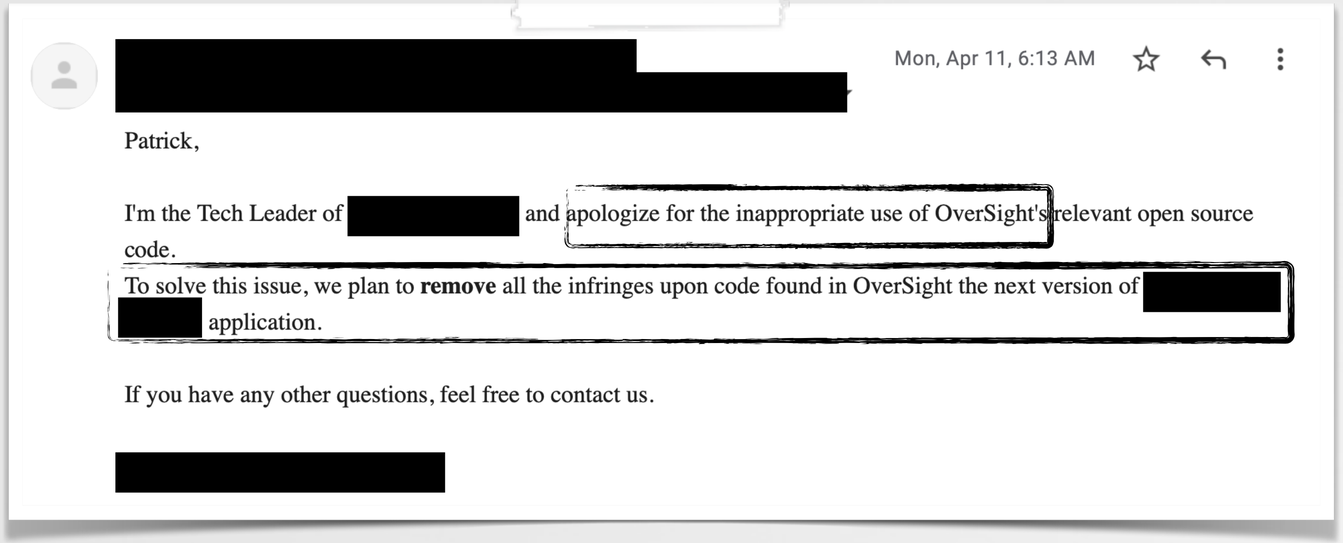
I am really sorry for this situation.

We are ready to take financial contribution as well.
I would like to add here our General Counsel [redacted] as he will be the right person to identify size of compensation and next steps there.
Please let me know if you have additional follow up questions.

3 Retroactive license (w/ compensation)

Another Win-Win Resolution

ack'd + code removal & financial compensation



[redacted] Mon, Apr 11, 6:13 AM

Patrick,

I'm the Tech Leader of [redacted] and apologize for the inappropriate use of OverSight's relevant open source code.

To solve this issue, we plan to **remove** all the infringes upon code found in OverSight the next version of [redacted] application.

If you have any other questions, feel free to contact us.

[redacted]

1 **Acknowledgement**

2 **Code Removal**

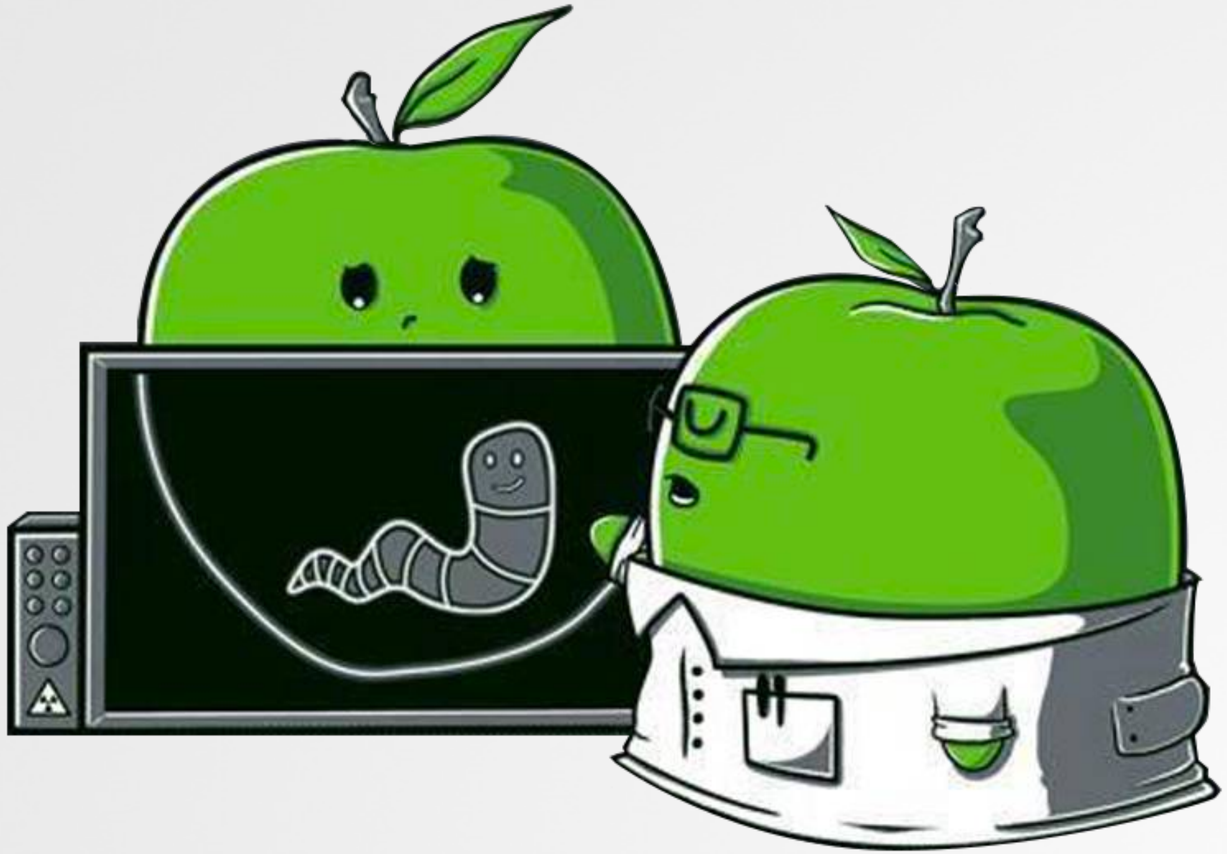
3 **Financial compensation**

Overall the product process did not meet our high standards, and we can and we will do better in the future. So after talking internally, we are more than happy to [redacted] donation

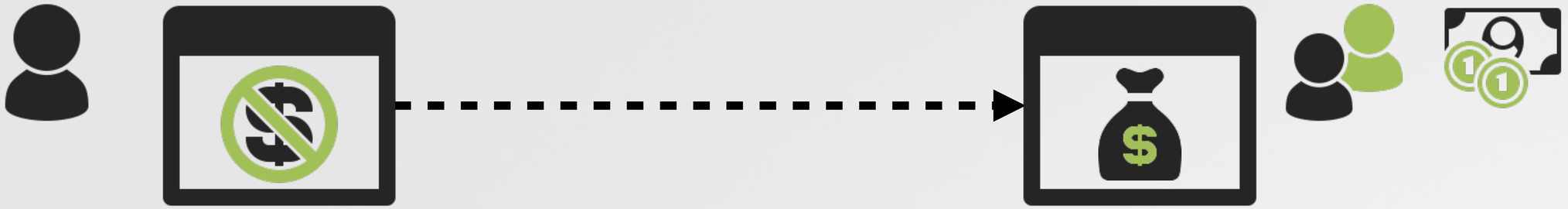
We will treat this as a learning experience. We are an active participant in the open source community and a firm believer in open source and intellectual property. We value you and your foundation's contribution to that community.

Conclusions

+takeaways



For Developers



Assume your code will be stolen
(regardless if it's closed or open source).



Hunt, confirm, approach
...and resolve (+recover reparations)

For Corporations

tl;dr don't steal other ppl algorithms for commercial gain

1. Educate your employees (developers)



legal issues



"optics"

> Article 6 of the 1991 EU Computer Programs Directive allows reverse engineering for the purposes of interoperability, but prohibits it for the purposes of creating a competing product, and also prohibits the public release of information obtained through reverse engineering of software

2. Implement internal procedures (scans, etc.)
("Is this original code? Or where did you get it from?")



3. If approached, work to amicably resolve any issues

MAHALO !

"Friends of Objective-See"











CleanMyMac X









SmugMug



Guardian Mobile Firewall



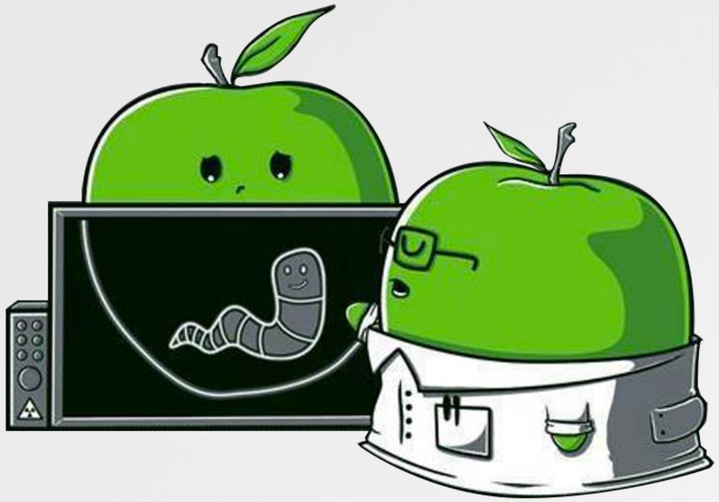
iVerify



Halo Privacy

Déjà Vu

Uncovering Stolen Algorithms in Commercial Products



RESOURCES:

"Zoom Zero Day: 4+ Million Webcams & maybe an RCE?"
https://infosecwriteups.com/zoom-zero-day-4-million-webcams-maybe-an-rce-just-get-them-to-visit-your-website-ac75c83f4ef5

"OverSight: Exposing Spies on macOS"
https://speakerdeck.com/patrickwardle/hack-in-the-box-2017-oversight-exposing-spies-on-macos

"Zero-Day TCC bypass discovered in XCSSET malware"
https://www.jamf.com/blog/zero-day-tcc-bypass-discovered-in-xcsset-malware/

Electronic Frontier Foundation
https://eff.org