New Mac backdoor using antiquated code

blog.malwarebytes.com/threat-analysis/2017/01/new-mac-backdoor-using-antiquated-code/

Thomas Reed

January 18, 2017

The first Mac malware of 2017 was brought to my attention by an IT admin, who spotted some strange outgoing network traffic from a particular Mac. This led to the discovery of a piece of malware unlike anything I've seen before, which appears to have actually been in existence, undetected, for some time, and which seems to be targeting biomedical research centers.

The malware was extremely simplistic on the surface, consisting of only two files:

```
~/.client
SHA256: ce07d208a2d89b4e0134f5282d9df580960d5c81412965a6d1a0786b27e7f044

~/Library/LaunchAgents/com.client.client.plist
SHA256: 83b712ec6b0b2d093d75c4553c66b95a3d1a1ca43e01c5e47aae49effce31ee3
```

The launch agent .plist file itself couldn't have been much simpler, simply keeping the .client running at all times.

```
<?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>KeepAlive</key>
 <true/>
 <key>Label</key>
 <string>com.client.client</string>
 <key>ProgramArguments</key>
 <array>
 <string>/Users/xxxx/.client</string>
 </array>
 <key>RunAtLoad</key>
 <true/>
 <key>NSUIElement</key>
 <string>1</string>
</dict>
</plist>
```

The .client file was where things got really interesting. It took the form of a minified and obfuscated perl script.

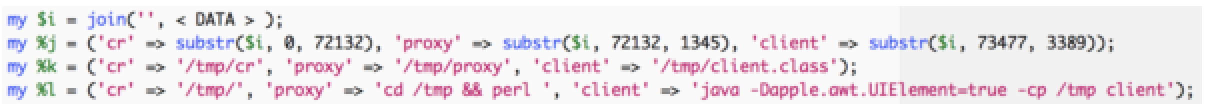
The perl script, among other things, communicates with the following command and control (C&C) servers:

```
99.153.29.240
eidk.hopto.org
```

The latter is a domain name managed by the dynamic DNS service no-ip.com.

The script also includes some code for taking screen captures via shell commands. Interestingly, it has code to do this both using the Mac “screencapture” command and the Linux “xwd” command. It also has code to get the system’s uptime, using the Mac “uptime” command or the Linux “cat /proc/uptime” command.

The most interesting part of the script can the found in the __DATA__ section at the end. Found there are a Mach-O binary, a second perl script and a Java class, which the script extracts, writes to the /tmp/ folder and executes. In the case of the Java class file, it is run with apple.awt.UIElement set to true, which means that it does not show up in the Dock.



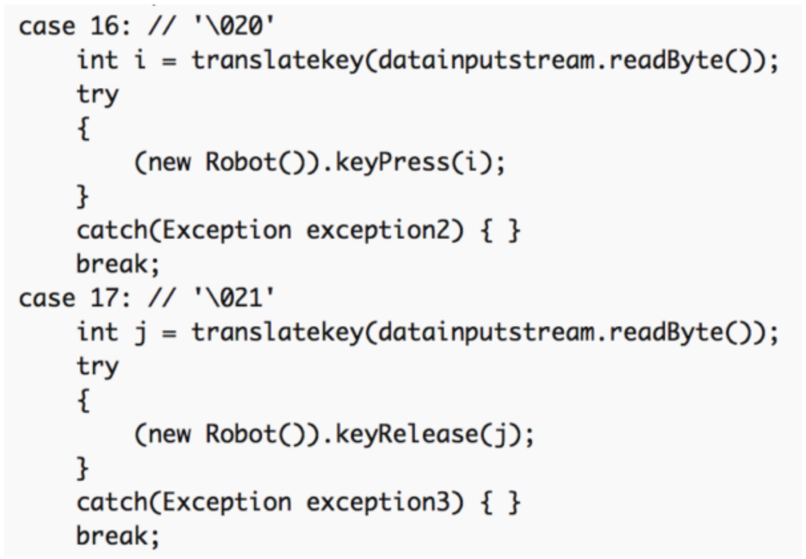
```
my $i = join('', < DATA > );
my %j = ('cr' => substr($i, 0, 72132), 'proxy' => substr($i, 72132, 1345), 'client' => substr($i, 73477, 3389));
my %k = ('cr' => '/tmp/cr', 'proxy' => '/tmp/proxy', 'client' => '/tmp/client.class');
my %l = ('cr' => '/tmp/', 'proxy' => 'cd /tmp && perl ', 'client' => 'java -Dapple.awt.UIElement=true -cp /tmp client');
```

The binary itself seems primarily interested in screen captures and webcam access, but interestingly, it uses some truly antique system calls for those purposes, such as:

```
SGGetChannelDeviceList
SGSetChannelDevice
SGSetChannelDeviceInput
SGInitialize
SGSetDataRef
SGNewChannel
QTNewGWorld
SGSetGWorld
SGSetChannelBounds
SGSetChannelUsage
SGSetDataProc
SGStartRecord
SGGetChannelSampleDescription
```

These are some truly ancient functions, as far as the tech world is concerned, dating back to pre-OS X days. In addition, the binary also includes the open source libjpeg code, which was last updated in 1998.

The Java class appears to be capable of receiving commands to do various tasks, which include yet another method of capturing the screen, getting the screen size and mouse cursor position, changing the mouse position, simulating mouse clicks, and simulating key presses. This component appears to be intended to provide a kind of rudimentary remote control functionality.



```
case 16: // '\020'
    int i = translatekey(datainputstream.readByte());
    try
    {
        (new Robot()).keyPress(i);
    }
    catch(Exception exception2) { }
    break;
case 17: // '\021'
    int j = translatekey(datainputstream.readByte());
    try
    {
        (new Robot()).keyRelease(j);
    }
    catch(Exception exception3) { }
    break;
```

We also observed the malware downloading a perl script, named “macsvc”, from the C&C server. This script uses mDNS to build a map of all the other devices on the local network, giving information about each device including its IPv6 and IPv4 addresses, name on the network and the port that is in use. It also appears to be making connection attempts to devices it finds on the network.

```
macsvc SHA256: b556c04c768d57af104716386fe4f23b01aa9d707cbc60385895e2b4fc08c9b0
```

Another file downloaded from the C&C server was named “afpscan”, and it seems to try to connect to other devices on the network.

```
afpscan SHA256: bbbf73741078d1e74ab7281189b13f13b50308cf03d3df34bc9f6a90065a4a55
```

The presence of Linux shell commands in the original script led us to try running this malware on a Linux machine, where we found that – with the exception of the Mach-O binary – everything ran just fine. This suggests that there may be a variant of this malware that is expressly designed to run on Linux, perhaps even with a Linux executable in place of the Mach-O executable. However, we have not found such a sample.

We were able to locate a couple Windows executable files on VirusTotal that communicate with the same C&C server. In addition, one contains strings that indicate that it uses the same libjpeg library from 1998 as the Mac Mach-O binary. Each of these samples were only ever submitted to VirusTotal once, in June and July of 2013, and are only detected by a few engines under generic names.

```
SHA256: 94cc470c0fdd60570e58682aa7619d665eb710e3407d1f9685b7b00bf26f9647
SHA256: 694b15d69264062e82d43e8ddb4a5efe4435574f8d91e29523c4298894b70c26
```

There are other indications that this malware has been circulating undetected for a long time. On one of the infected Macs, the launch agent file had a creation date in January of 2015. That’s not strong evidence of the true creation date, though, as those dates can easily be changed.

Further, there is a comment in the code in the macsvc file that indicates that a change was made for Yosemite (Mac OS X 10.10), which was released in October of 2014. This suggests that the malware has been around at least some time prior to Yosemite’s release.

```
if(/_(tcp|udp)\S*\s+(_\S+)$/){ $s="$2._$1"; }
elsif(/icloud\.com\.\s+(_[^\.]+\._(tcp|udp))\.\d+\.members\.btmm$/)
   { $s=$1; } # changed in yosemite
elsif(/icloud\.com\.\s+\.\s+_autotunnel6$/){ next; }
```

Another clue, of course, is the age of some of the code, which could potentially suggest that this malware goes back decades. However, we shouldn’t take the age of the code as too strong an indication of the age of the malware. This could also signify that the hackers behind it really don’t know the Mac very well and were relying on old documentation. It could also be that they’re using old system calls to avoid triggering any kind of behavioral detections that might be expecting more recent code.

Ironically, despite the age and sophistication of this malware, it uses the same old unsophisticated technique for persistence that so many other pieces of Mac malware do: a hidden file and a launch agent. This makes it easy to spot, given any reason to look at the infected machine closely (such as unusual network traffic). It also makes it easy to detect and easy to remove.

The only reason I can think of that this malware hasn't been spotted before now is that it is being used in very tightly targeted attacks, limiting its exposure. There have been a number of stories over the past few years about Chinese and Russian hackers targeting and stealing US and European scientific research. Although there is no evidence at this point linking this malware to a specific group, the fact that it's been seen specifically at biomedical research institutions certainly seems like it could be the result of exactly that kind of espionage.

Malwarebytes will detect this malware as OSX.Backdoor.Quimitchin. (Why the name? Because the quimitchin were Aztec spies who would infiltrate other tribes. Given the "ancient" code, we thought the name fitting.) Apple calls this malware Fruitfly and has released an update that will be automatically downloaded behind the scenes to protect against future infections.