# Detecting XLoader | A macOS ‘Malware-as-a-Service’ Info Stealer and Keylogger

By Phil Stokes

Published: 2021-07-26 · Archived: 2026-04-05 21:15:58 UTC

Threat actors have come to recognize the reality that today’s organizations operate fleets of devices encompassing all the major OS vendors – Apple, Microsoft, Google and many flavors of Linux – and are adapting accordingly. Threats that can be compiled on one platform but produce executables targeting many are a productivity boon to criminals, who now operate in an increasingly competitive environment trying to sell their wares.

The latest such threat to come to attention is XLoader, a Malware-as-a-Service info stealer and keylogger that researchers say was developed out of the ashes of FormBook. Unlike its Windows-only predecessor, XLoader targets both Windows and macOS. In this post, we take an initial look at the macOS version of XLoader, describe its behavior and show how XLoader can be detected on Apple’s Mac platform.


## XLoader for Mac – Java Runtime For the Steal

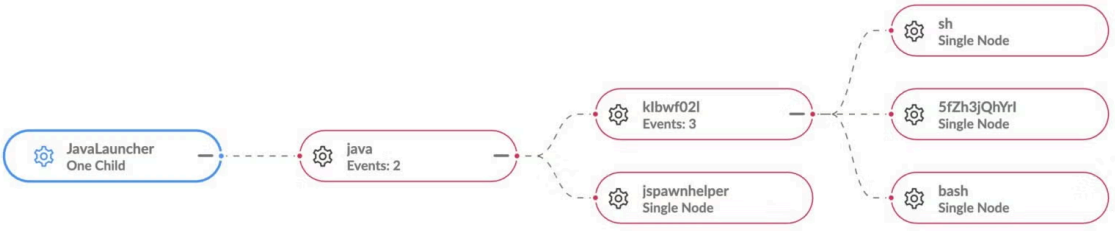
The macOS sample we analyzed comes as both a standalone binary and as a compiled `.jar` file. The `.jar` file appears to be distributed as an attachment in a phishing lure, such as in this document `Statement SKBMT 09818.jar`.


XLoader is likely distributed by mail spam

Such files require the Java Runtime Environment, and for that reason the malicious `.jar` file will not execute on a macOS install out of the box, since Apple stopped shipping JRE with Macs over a decade ago.

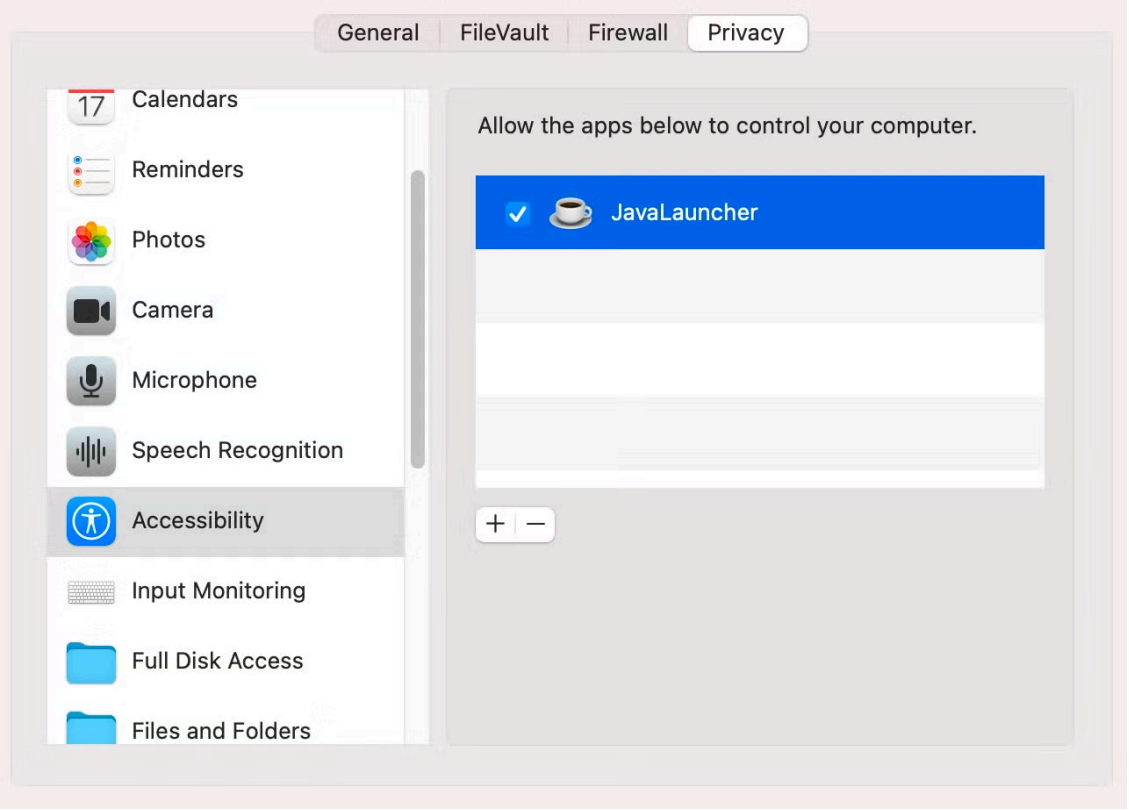
Nonetheless, Java is still a common requirement in enterprise environments and is still in use for some banking applications. As a result, many organizations will have users that either do or must install the Oracle version of Java to meet these needs. As a 3rd party plugin, the Oracle JRE is installed at `/Library/Internet Plug-Ins/JavaAppletPlugin.plugin`.

When the malware is executed as a `.jar` file, the execution chain begins with the OS-provided JavaLauncher at `/System/Library/CoreServices/JavaLauncher.app`.




XLoader's execution chain begins with the JavaLauncher

The JavaLauncher is also populated in the Accessibility pane in **System Preferences' Privacy** tab and a dialog is popped requesting the user to grant access for automation. As we shall see below, this is likely leveraged as part of the info stealer's functionality.

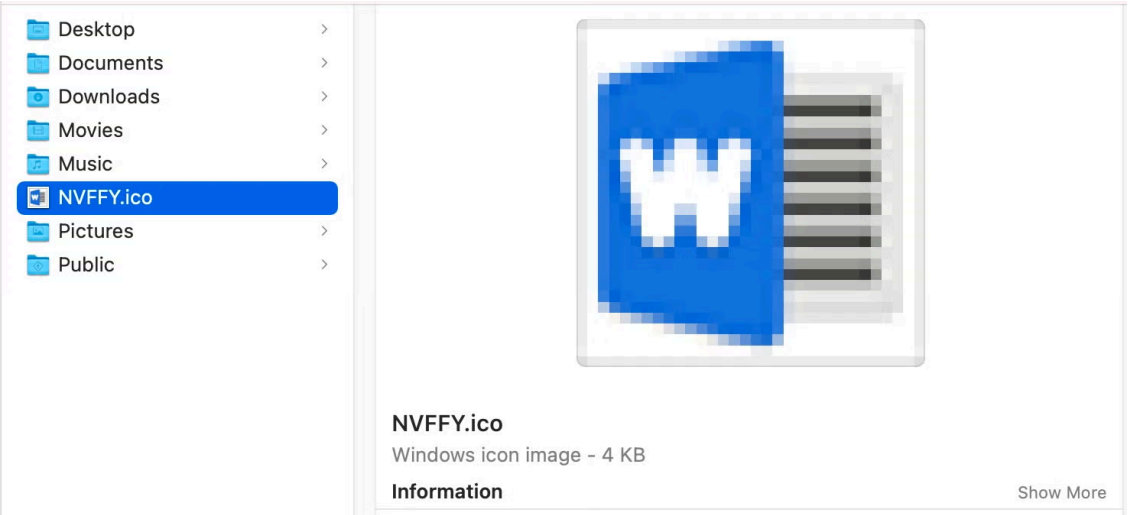



The JavaLauncher requests access to control other applications

The `com.oracle.JavaInstaller` will also populate the 'Full Disk Access' table in the same tab. This remains unchecked by default and, at least on our test, no dialog was presented to the user to request permissions.

## XLoader Behavior on macOS

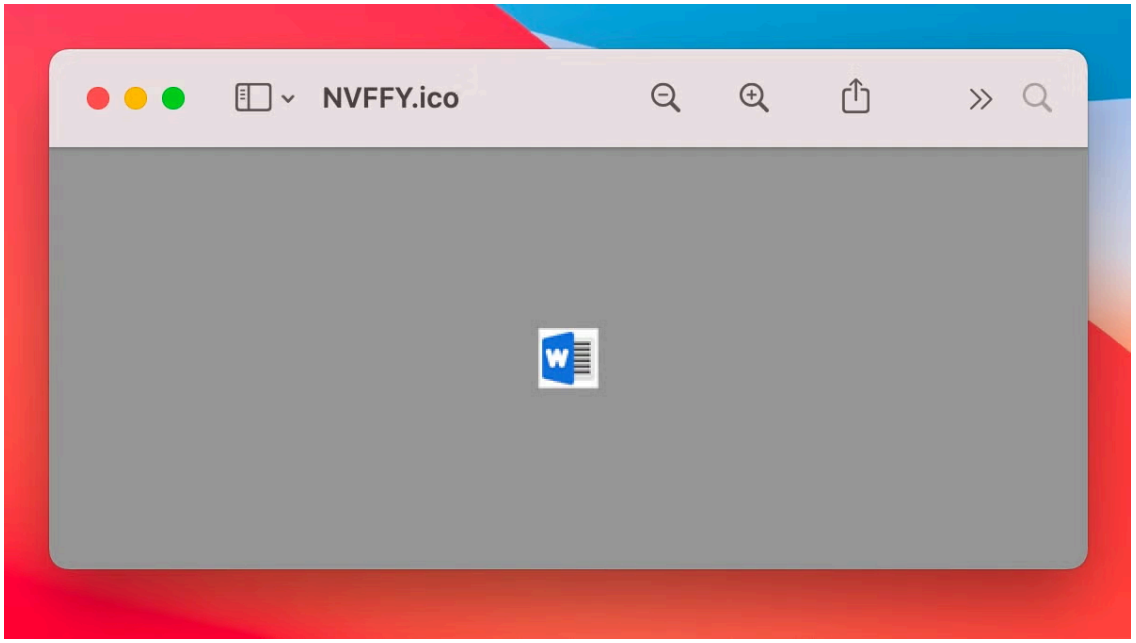
On execution the malware drops a 32×32 pixel Windows image file in the user's home directory called `NVFFY.ico`.




A Windows icon file is dropped in the user's home folder

The user's default image viewer – typically the built-in Preview.app – will be launched to display this image. At this point, one could imagine that even the most unsuspecting user opening the 'Statement SKBT' file is going to

think that something is amiss.




The .ico file as presented to the victim

It's unclear what the malware authors were thinking here: perhaps the sample is an early development or a test sample. Alternatively, this may be a reflection of the hazards of cross-platform malware, where the author's assumptions on the Windows platform were not fully tested on a macOS device.

In any case, no interaction is required from the user and the malware continues to drop and execute the rest of its components. This involves dropping and executing a Mach-O file in the user's Home folder. This file, `kIbwf02l`, writes a hidden application bundle, also located in the victim's Home folder, and containing a copy of itself. It then writes and loads a user LaunchAgent with a program argument pointing to the copy in the hidden app bundle. From then on, the `kIbwF02l` file appears to be redundant but is not cleaned up by the malware.

```
drwxr-xr-x   3 macthreats  staff    96 23 Jul 13:01 .
drwx------@ 64 macthreats  staff  2048 23 Jul 13:07 ..
-rw-------   1 macthreats  staff   481 23 Jul 13:01 com.URzH.5fZh3jQhYrI.plist
macthreats@macos-detection-lab LaunchAgents % ls -l@ com.URzH.5fZh3jQhYrI.plist
-rw-------  1 macthreats  staff  481 23 Jul 13:01 com.URzH.5fZh3jQhYrI.plist
macthreats@macos-detection-lab LaunchAgents % plutil -p com.URzH.5fZh3jQhYrI.plist
{
  "KeepAlive" => 0
  "Label" => "com.URzH.5fZh3jQhYrI"
  "ProgramArguments" => [
    0 => "/Users/macthreats/.URzH/5fZh3jQhYrI.app/Contents/MacOS/5fZh3jQhYrI"
    1 => "start"
  ]
  "RunAtLoad" => 1
}
macthreats@macos-detection-lab LaunchAgents %
```

Example of an XLoader LaunchAgent

The label for the LaunchAgent and the names of the hidden app and executable are all randomized and vary from execution to execution. The binary is passed the argument `start` as a launch parameter.

The hidden application is itself a barebones bundle containing only the Info.plist and the Mach-O executable.

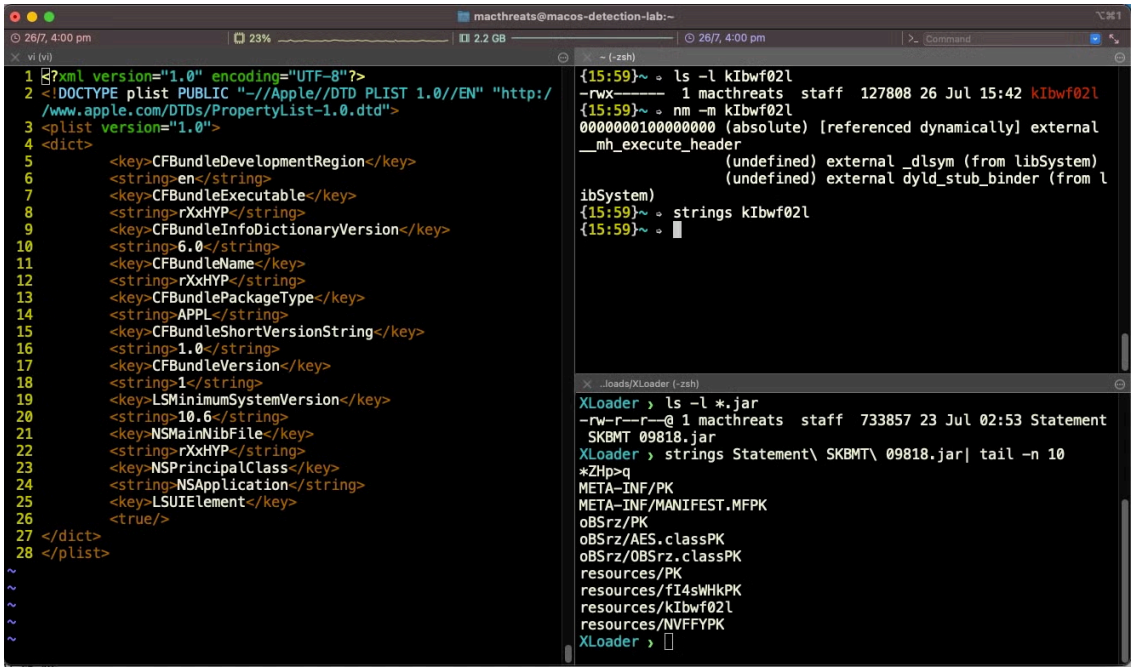
```
xphil@sentinel-macos-11 vRoh8.app % cd Contents
xphil@sentinel-macos-11 Contents % ls -al
total 8
drwxr-xr-x  4 xphil  staff  128 Jul 23 15:11 .
drwxr-xr-x  3 xphil  staff   96 Jul 23 15:11 ..
-rw-------  1 xphil  staff  780 Jul 23 15:11 Info.plist
drwxr-xr-x  2 xphil  staff   64 Jul 23 15:18 MacOS
xphil@sentinel-macos-11 Contents % pwd
/Users/xphil/.9nphXvrphLBH/vRoh8.app/Contents
```

XLoader's hidden application bundle

A copy of the same executable, *sans* bundle and with the filename `kIbwf02l`, is also dropped in the User's home directory.

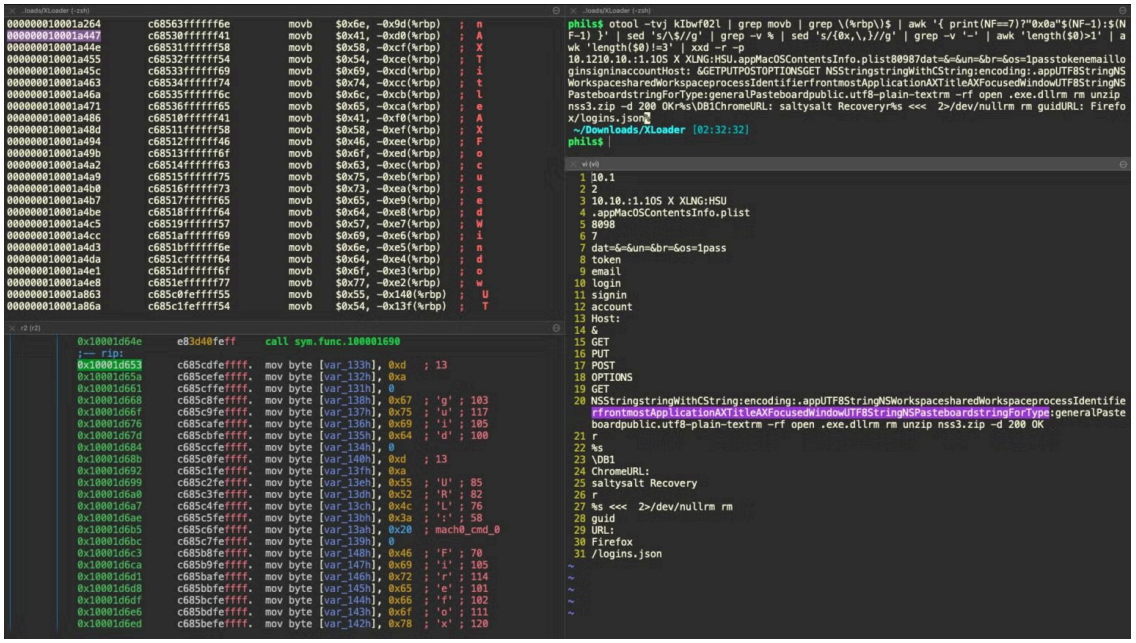
## Analysis of the XLoader Mach-O

The compiled Mach-O executable pointed to by the persistence agent is heavily stripped and obfuscated. As the image below indicates, static analysis using tools like strings will show little, and dynamic analysis is complicated by a number of anti-debugging features.




Left: the hidden app's Info.plist. Right: strings and symbols in the executables

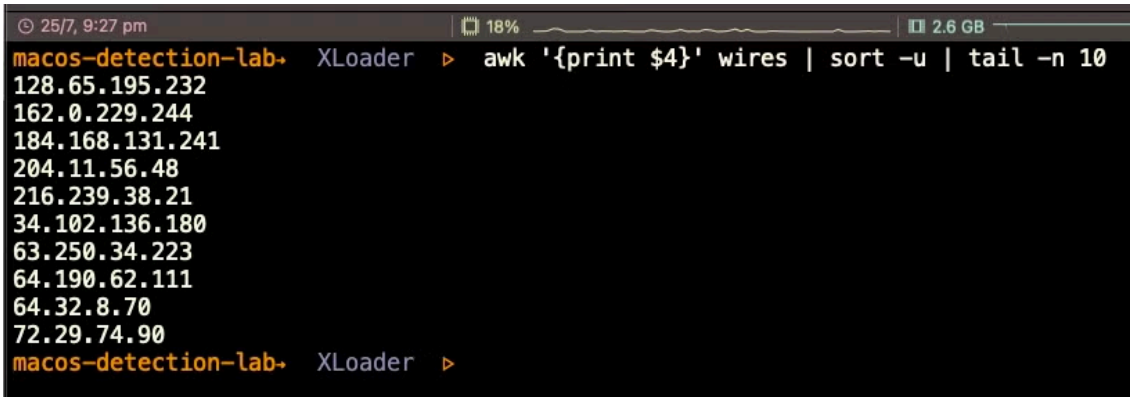
For the purposes of quick triage, we extracted the stackstrings from the Mach-O using otool to get an initial idea of the info stealer's functionality. With further processing either manually or with radare2, we can match these strings to particular functions.



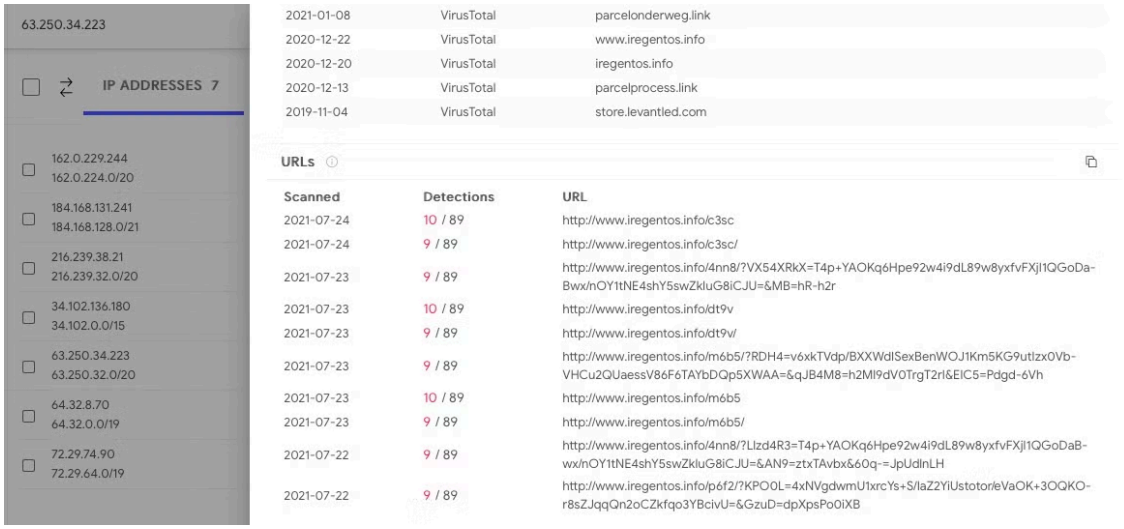
Stack strings found in XLoader's macOS version

The strings here show that XLoader attempts to steal credentials from Chrome and Firefox browsers. We also see an indication that the malware calls the `NSWorkspace` API to identify the front window via the Accessibility API `AXTitleFocusedWindow` and leverages `NSPasteboard`, likely to copy information from the window of the user's currently active process. Calling Accessibility APIs requires user consent as this functionality is controlled by TCC. As noted above, the JavaLauncher has such permissions.

Other researchers have suggested that XLoader's internet traffic is laden with decoys to disguise the actual C2 used to transmit data. As we did not observe any credential stealing traffic in our test, we cannot confirm that suspicion, but XLoader's internet traffic is certainly 'noisy'. We observed the malware reaching out to a variety of known phishing and malware sites.




Some of the IP addresses contacted by the XLoader malware



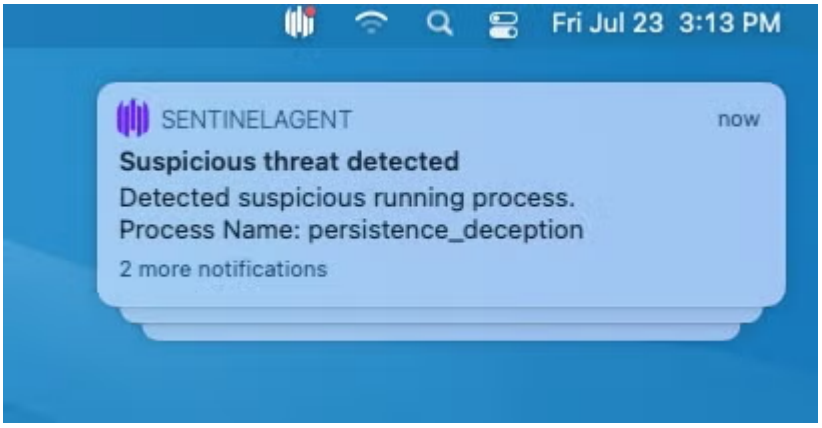

One of a number of malicious domains XLoader contacts (VirusTotal)

## Detecting XLoader Infostealer on macOS

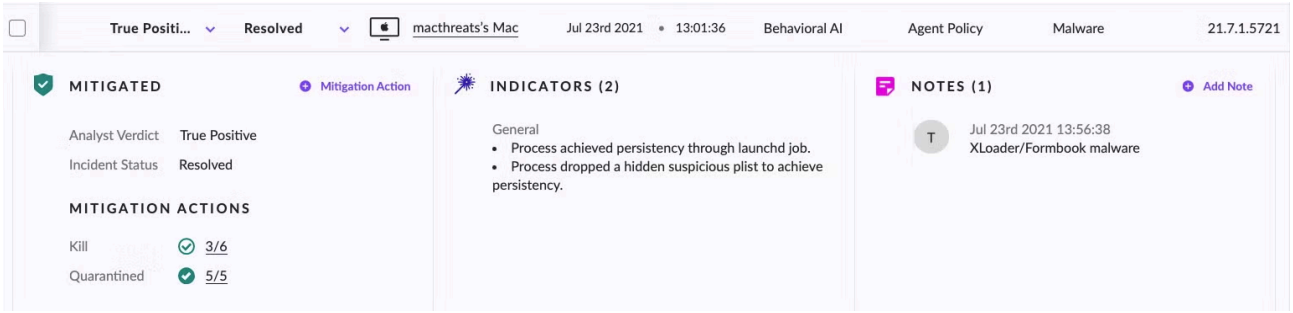
At the end of this post we provide a number of macOS-specific Indicators of Compromise to help organizations and users in general identify an XLoader infection. SentinelOne customers are protected against this malware automatically, regardless of whether it is executed via the Java Runtime Environment or by the standalone XLoader Mach-O.

In our test, we set the agent to ‘Detect-only’ policy in order to observe the malware’s behaviour. Customers are advised to always use the ‘Protect’ policy which prevents execution of malware entirely.

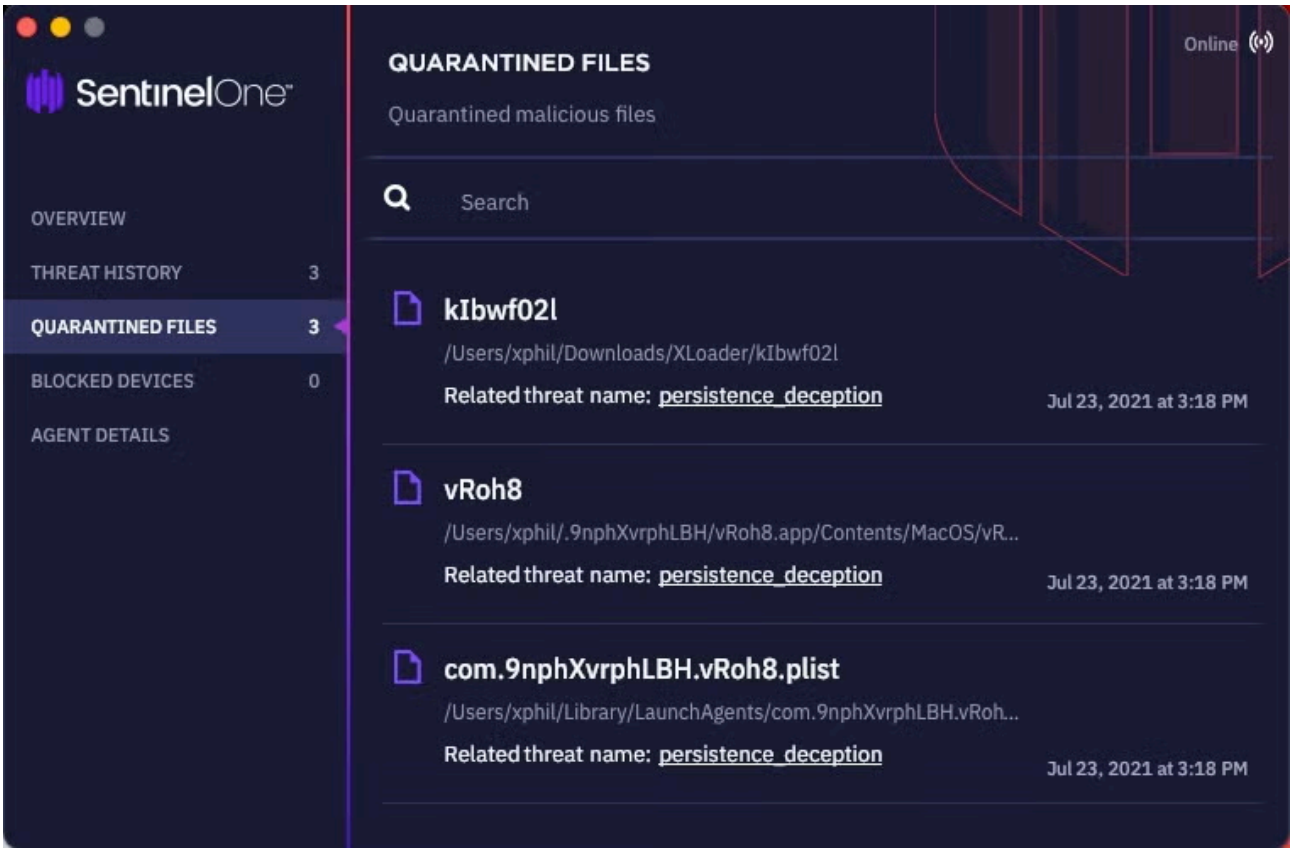
In ‘Detect-only’ mode, the target’s Mac device will immediately alert the user via Notifications:




Security teams and IT administrators, meanwhile, would see something similar to the following in the Management console.




After remediation, the UI (version 21.7EA) on the device indicates that the threat has been successfully killed and quarantined.




## Conclusion

XLoader is an interesting and somewhat unusual example in the macOS malware world. It's dependency on Java and its functionality suggests it is primarily targeting organizations where the threat actors expect Java applications to be in use. Among other things, that includes certain online banking applications, and the attractiveness from a criminal's perspective of a keylogger and info stealer in that environment can certainly be understood. It is also worth noting that the malware's minimum system requirement is 10.6 Snow Leopard (over 10 years old), so the author's are certainly casting their net wide. On the other hand, the implementation on macOS is clumsy at best and is likely to raise suspicions. No doubt the malware authors will be looking to improve on this in future iterations.

## Indicators of Compromise

### SHA1 Hashes

XLoader Mach-O Executable: KIbwf02l
7edead477048b47d2ac3abdc4baef12579c3c348

Suspected Phishing lure attachment: Statement SKBMT 09818.jar
b8c0167341d3639eb1ed2636a56c272dc66546fa

Example Persistence LaunchAgent: com.j85H64iPLnW.rXxHYP
cb3e7ac4e2e83335421f8bbc0cf953cb820e2e27

### Contacted IPs

128.65.195.232
162.0.229.244
184.168.131.241
204.11.56.48
216.239.38.21
34.102.136.180
63.250.34.223
64.190.62.111
64.32.8.70
72.29.74.90

### Interesting Strings

```
.appMacOSContentsInfo.plist
.exe.dll
/logins.json
10.:1.1OS X XLNG:
200 OK
80987dat=&=&un=&br=&os=1
DB1ChromeURL:
guidURL: Firefox
NSStringstringWithCString:encoding:
open
passtokenemailloginsigninaccountHost: &GETPUTPOSTOPTIONSGET
r%s <<<  2>/dev/null
Recovery
rm -rf
rm unzip nss3.zip -d
saltysalt
UTF8StringNSPasteboardstringForType:generalPasteboardpublic.utf8-plain-text
UTF8StringNSWorkspacesharedWorkspaceprocessIdentifierfrontmostApplicationAXTitleAXFocusedWindow
```

Source: https://www.sentinelone.com/blog/detecting-xloader-a-macos-malware-as-a-service-info-stealer-and-keylogger/