How SysJoker and DazzleSpy Malware Target macOS

By Phil Stokes

Published: 2022-02-01 · Archived: 2026-04-10 02:13:24 UTC

As last year closed out, we provided a round up of the previous 12 months of Mac malware, making the observation that, among other things, 2021's macOS malware cohort saw a focus on spyware and the targeting of users in Asia, particularly China and Hong Kong. The first month of 2022 has seen those trends continue with two new malware campaigns discovered in January, namely SysJoker and DazzleSpy.

In this post, we give brief overviews of these two new malware families, offering both additional details not previously reported along with indicators for detection and threat hunting.

SysJoker (11th Jan, 2022)

The first new Mac malware report of 2022 came courtesy of researchers at Intezer in the form of a threat they dubbed SysJoker, which comes in Windows, Linux and macOS variants. Researchers say that the Linux version was found in-the-wild infecting a server belonging to "a leading educational institution".

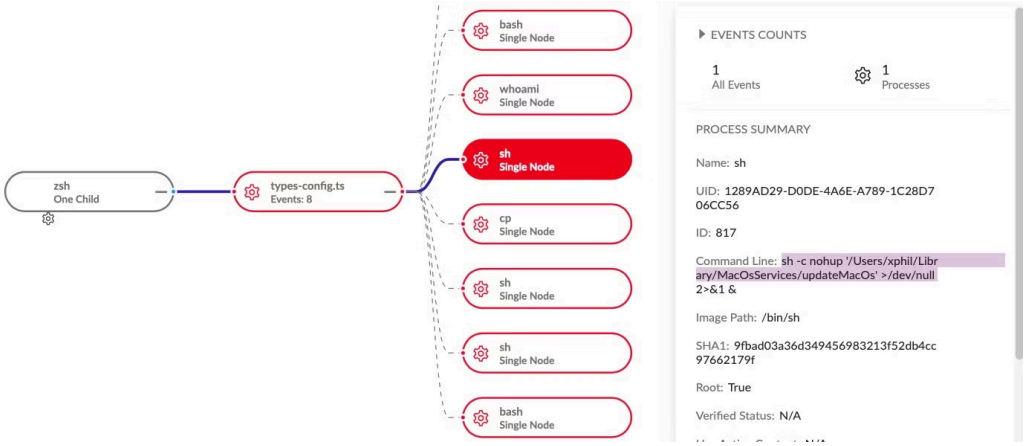
The Mac-specific variant of this malware is a Universal binary named `types-config.ts`, compiled for both Intel x86 and Apple silicon M1 arm64 architectures.

Upon execution, the Mach-O installs a persistence LaunchAgent that masquerades as an Apple launch service `~/Library/LaunchAgents/com.apple.update.plist`.

```
<?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>com.apple.update</string>
        <key>LimitLoadToSessionType</key>
        <string>Aqua</string>
        <key>ProgramArguments</key>
        <array>
                <string>/Users/xphil/Library/MacOsServices/updateMacOs</string>
        </array>
<key>KeepAlive</key>
        <dict>
           <key>SuccessfulExit</key>
           <true/>
        </dict>
        <key>RunAtLoad</key>
        <true/>
</dict>
</plist>
```

Persistence mechanism used by SysJoker malware on macOS

The fake service targets an executable called `~/Library/MacOsServices/updateMacOs`. This file is also written by the `types-config.ts` file and is in fact a straight copy of itself. The SentinelOne agent captures the chain of execution and displays it in the Management console for easy pivoting and threat hunting.



OSX.SysJoker backdoor execution chain as captured by the SentinelOne agent

The malware is written in C++ and much of the initial action occurs in the `entry.init0` function. Using r2, we can get a quick summary of the function's important strings.

```
[0x10001399a]> s entry.init0
[0x10001399a]> afns
0x1000139e2 0x100015f3d str.MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDkfNlSe7jm7sGSrSSUpV3HUl3vEwuhxn4qBY6aRFL91x0HIgcH2AM2rOllLdoV8v1vtG1oPt9QpC1jSxShnFw8evGrYnqaou7gLsY5J2B06eq5UW7OXgb77WNbU90vyUbZAucfzy0eF1HqtBNbkXiQ6SSbquuvFPUepqUEjUSQIDAQAB
0x100013a06 0x10001602b str.updateMacOs
0x100013a23 0x100016037 str.whoami
0x100013a73 0x10001603e str._Users_
0x100013a89 0x100016046 str._Library_MacOsServices
0x100013ae4 0x10001603e str._Users_
0x100013afe 0x10001605d str._Library_SystemNetwork
0x100013bfa 0x100016076 str.https:__drive.google.com_uc_exportdownloadid1W64PQQxrwY3XjBnv_QAeBQu_ePr537eu
[0x10001399a]> _
```

Some of the embedded strings in the SysJoker binary

The "drive.google.com" address delivers a file "domain.txt" that contains an obfuscated domain name address. The key shown above at address `0x1000139e2` is used to decode the contents of "domain.txt", which turns out to to be the DNS address "graphic-updater.com".

Other hardcoded strings are then concatenated with the decoded DNS address to form a full C2.

```
https:
```

```
[0x100016496]> izz~api:0
646  0x00016496 0x100016496 11  12   5.__TEXT.__cstring       ascii   /api/attach
[0x100016496]> axt @@=`izz~api:0[2]`
sym.func.10000503c 0x1000057e5 [DATA] lea rdx, str._api_attach
main 0x100006d89 [DATA] lea rdx, str._api_attach
main 0x10000738b [DATA] lea rdx, str._api_attach
[0x100016496]> _
```

The C2 address is determined on-the-fly during execution

We note that SysJoker has a peculiarity that, to our knowledge, has not been described by other researchers. In our tests, if the malware is run as root when the path

```
/Users/root/Library/SystemNetwork
```

does not exist, the malware will abort.

That's an unusual path, as the root user on macOS typically exists under `/var/root`, not `/Users/root`.

Whether this is an oversight or a peculiarity of SysJoker's intended target is unclear. At this point, we have no explanation for this behaviour, but merely note that if `/Users/root` does exist, then the malware executes as expected, and drops the components under that file path hierarchy.

```
sh-3.2# cd /Users/root
sh-3.2# ls -lR
total 0
drwxr-xr-x  5 root  admin  160  1 Feb 21:52 Library

./Library:
total 0
drwxr-xr-x  3 root  admin  96  1 Feb 21:52 LaunchAgents
drwxr-xr-x  3 root  admin  96  1 Feb 21:52 MacOsServices
drwxr-xr-x  2 root  admin  64  1 Feb 21:51 SystemNetwork

./Library/LaunchAgents:
total 8
-rw-r--r--  1 root  admin  579  1 Feb 21:52 com.apple.update.plist

./Library/MacOsServices:
total 328
-rwxr-xr-x@ 1 root  admin  164624  1 Feb 21:52 updateMacOs

./Library/SystemNetwork:
sh-3.2#
```

SysJoker uses an unorthodox path for a macOS root user

According to previous researchers who also analyzed the Windows and Linux variants, SysJoker's primary purpose is to await commands from the C2. We, and our sample, did indeed wait, but the C2 appeared to be uninterested in talking to either of us. Intezer has more details on the backdoor's functionality.

How To Protect Against OSX.SysJoker

The SentinelOne Singularity platform fully detects OSX.SysJoker.

SentinelOne detects SysJoker on execution

Aside from the one reported in-the-wild incident against a “leading educational institution”, it is unclear at this time how SysJoker is distributed, who it targets, or what the authors’ objectives are. However, the cross-platform nature of the malware suggests that it may be part of a wider campaign, and it is imperative that organizations have a capable multi-engined security solution in place to defend against these kinds of attacks.

DazzleSpy (25th Jan)

OSX.DazzleSpy was discovered by ESET researchers following the same trail as Google’s Project Zero from a poisoned watering hole targeting Hong Kong pro-democracy activists. Whereas Google’s investigation led them to macOS.Macma, researchers Marc L’Etienne and Anton Cherepanov caught a quite different payload.

OSX.DazzleSpy comes in the form of an unsigned, Mach-O file compiled for Intel x86 architecture, although it’s perfectly possible that undiscovered ARM versions exist as well.

On execution, the Mach-O installs a persistence LaunchAgent that masquerades as an Apple launch service at `~/Library/LaunchAgents/com.apple.softwareupdate`. This fake service targets an executable called “softwareupdate” written inside a hidden folder of the user’s home folder, `~/.local/softwareupdate`.

```
<?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>KeepAlive</key>
        <true/>
        <key>Label</key>
        <string>com.apple.softwareupdate</string>
        <key>ProgramArguments</key>
        <array>
                <string>/Users/xphil/.local/softwareupdate</string>
                <string>1</string>
        </array>
        <key>RunAtLoad</key>
        <true/>
        <key>SuccessfulExit</key>
        <true/>
</dict>
</plist>
```

DazzleSpy LaunchAgent property list for persistence

The executable “softwareupdate” contains a mixture of public and private frameworks. On the public side, the malware authors have adopted the tonymillion Reachability framework to determine network connections, YYModel for efficient parsing of JSON data, and GCDAsyncSocket to handle TCP/IP socket networking tasks. A date comparison method, `+(int)compareOneDay:(NSDate *)oneDay withAnotherDay:(NSDate *)anotherDay`, also appears to have been lifted from a Chinese-language programming forum.

```
0x100024a30    7 588          method.class.MethodClass.compareOneDay:withAnotherDay:
:> izz~compare
621  0x0005f6ba 0x10005f6ba 8    9    3.__TEXT.__objc_methname  ascii   compare:
624  0x0005f6ee 0x10005f6ee 29   30   3.__TEXT.__objc_methname  ascii   compareOneDay:withAnotherDay:
3250 0x00096db6 0x100096db6 44   45                             ascii   +[MethodClass compareOneDay:withAnotherDay:]
4745 0x000a50b6 0x1000a50b6 44   45                             ascii   +[MethodClass compareOneDay:withAnotherDay:]
```

```
e8cb830300     call sym.imp.objc_alloc    ;[2]
488b35140705.  mov rsi, qword [section.22.__DATA.__objc_selrefs]    ; [0x1000751a0:8]=0x10005d860 section.3.__TEXT.__objc_methname ; "`\xd8\x0
4889c7         mov rdi, rax
ff15b3380400   call qword [reloc.objc_msgSend] ;[3] ; [0x100068348:8]=0
488d0d145f04.  lea rcx, str.cstr.MM_yyyy    ; 0x10006a9b0
488945d0       mov qword [var_30h], rax
488b45d0       mov rax, qword [var_30h]
488b35050e05.  mov rsi, qword [0x1000758b0]    ; [0x1000758b0:8]=0x10005f587 str.setDateFormat:
4889c7         mov rdi, rax
4889ca         mov rdx, rcx
ff1591380400   call qword [reloc.objc_msgSend] ;[3] ; [0x100068348:8]=0
488d05f25d04.  lea rax, str.cstr.Asia_Shanghai    ; 0x10006a8b0
488b0dd31b05.  mov rcx, qword [reloc.NSTimeZone]    ; [0x100076698:8]=0
488b35d40d05.  mov rsi, qword [0x1000758a0]    ; [0x1000758a0:8]=0x10005f568 str.timeZoneWithName:
4889cf         mov rdi, rcx
4889c2         mov rdx, rax
ff1570380400   call qword [reloc.objc_msgSend] ;[3] ; [0x100068348:8]=0
```

DazzleSpy contains a mix of public and private frameworks and methods

For functionality, DazzleSpy contains code for searching and writing files, exfiltrating environmental info, dumping the keychain, running a remote desktop and running shell commands, among others.

```
[0x100022569]> axt 0x00000001000049c0
method.MethodClass.getAllhardwareports 0x100022ece [CODE] mov rsi, qword [method.class.Exec.doShellInCmd:]
method.MethodClass.clearTrace 0x100023601 [CODE] mov rsi, qword [method.class.Exec.doShellInCmd:]
method.MethodClass.doShellInCmd: 0x100024cb6 [CODE] mov rsi, qword [method.class.Exec.doShellInCmd:]
method.class.Singleton.installDaemon 0x1000369c8 [CODE] mov rsi, qword [method.class.Exec.doShellInCmd:]
method.class.Singleton.installDaemon 0x100036d37 [CODE] mov rsi, qword [method.class.Exec.doShellInCmd:]
method.Singleton.analysisData:Socket: 0x100039cbb [CODE] mov rsi, qword [method.class.Exec.doShellInCmd:]
method.Singleton.analysisData:Socket: 0x10003eb83 [CODE] mov rsi, qword [method.class.Exec.doShellInCmd:]
method.Singleton.analysisData:Socket: 0x10003edeb [CODE] mov rsi, qword [method.class.Exec.doShellInCmd:]
method.AutoupdateClassObject.updateCommaindRemotePath: 0x10005418d [CODE] mov rsi, qword [method.class.Exec.doShellInCmd:]
method.KeychainClassObject.getPass:cmdTo: 0x100057596 [CODE] mov rsi, qword [method.class.Exec.doShellInCmd:]
method.KeychainClassObject.getKeychain: 0x10005797f [CODE] mov rsi, qword [method.class.Exec.doShellInCmd:]
method.class.KeychainClassObject.unzipFile:toPath: 0x10005b2a0 [CODE] mov rsi, qword [method.class.Exec.doShellInCmd:]
```

A number of methods are run as shell commands via NSTask APIs

DazzleSpy collects and drops a number of other files in the hidden `~/.local` directory related to espionage and data collection.

```
1511 0x00064b41 0x100064b41 30   31   4.__TEXT.__cstring        ascii   .local/security/keystealDaemon
1563 0x00064d5d 0x100064d5d 19   20   4.__TEXT.__cstring        ascii   .local/security.zip
1564 0x00064d71 0x100064d71 24   25   4.__TEXT.__cstring        ascii   %@/.local/softwareupdate
1628 0x0006508a 0x10006508a 21   22   4.__TEXT.__cstring        ascii   %@/.local/SearchFiles
1710 0x00065789 0x100065789 23   24   4.__TEXT.__cstring        ascii   %@/.local/RecoveryFiles
1745 0x000659dd 0x1000659dd 15   16   4.__TEXT.__cstring        ascii   .local/security
```

Some of the hardcoded paths found in the DazzleSpy executable

```
~/.local/softwareupdate
~/.local/security/keystealDaemon
~/.local/security.zip
~/.local/SearchFiles
~/.local/RecoveryFiles
~/.local/security
```

Although we only saw the first of these files dropped in our tests, analysis of the static code suggests that another hidden directory, `.Documenty`, may also be used by the malware.

```
1739 0x0006590a 0x10006590a 28   29   4.__TEXT.__cstring        ascii   .Documenty/security/keys.err
1740 0x00065929 0x100065929 37   38   4.__TEXT.__cstring        ascii   .Documenty/security/security-unsigned
1742 0x00065977 0x100065977 43   44   4.__TEXT.__cstring        ascii   .Documenty/security/libkeystealClient.dylib
1743 0x000659a9 0x1000659a9 34   35   4.__TEXT.__cstring        ascii   .Documenty/security/keystealDaemon
```

A path we didn't see on execution, but potentially useful for hunting

The authors appear to have been careless (or perhaps deliberate!) in leaving artifacts from the development environment. As noted by ESET, one user name embedded in the malware is "wangping", but we also note two others: "wp" and "XpathX".

```
ascii   /Users/XpathX/Library/Cookies
ascii   /Users/wp/aa.txt
ascii   /Users/wp/bb.txt
ascii   /Users/wangping/pangu/create_source/poke/osxrk_commandLine/
```

Usernames found embedded in the DazzleSpy binary

Of these, “XpathX” seems to have a number of paths typical of an active user, but why these should have found their way into the code is both mysterious and suspicious.

```
1674 0x00065284 0x100065284 29   30   4.__TEXT.__cstring       ascii   /Users/XpathX/Pictures/Photos
1675 0x000652a2 0x1000652a2 21   22   4.__TEXT.__cstring       ascii   /Users/XpathX/Desktop
1676 0x000652b8 0x1000652b8 52   53   4.__TEXT.__cstring       ascii   /Users/XpathX/Library/Application Support/MobileSync
1677 0x000652ed 0x1000652ed 65   66   4.__TEXT.__cstring       ascii   /Users/XpathX/Library/Application Support/CallHistoryTransactions
1678 0x0006532f 0x10006532f 52   53   4.__TEXT.__cstring       ascii   /Users/XpathX/Library/Application Support/com.apple*
1679 0x00065364 0x100065364 51   52   4.__TEXT.__cstring       ascii   /Users/XpathX/Library/Application Support/Knowledge
1680 0x00065398 0x100065398 54   55   4.__TEXT.__cstring       ascii   /Users/XpathX/Library/Application Support/FileProvider
1681 0x000653cf 0x1000653cf 51   52   4.__TEXT.__cstring       ascii   /Users/XpathX/Pictures/Photos Library.photoslibrary
1682 0x00065403 0x100065403 39   40   4.__TEXT.__cstring       ascii   /Users/XpathX/Library/Caches/com.apple*
1683 0x0006542b 0x10006542b 20   21   4.__TEXT.__cstring       ascii   /Users/XpathX/.Trash
1684 0x00065440 0x100065440 23   24   4.__TEXT.__cstring       ascii   /Users/XpathX/Documents
1685 0x00065458 0x100065458 23   24   4.__TEXT.__cstring       ascii   /Users/XpathX/Downloads
1686 0x00065470 0x100065470 38   39   4.__TEXT.__cstring       ascii   /Users/XpathX/Library/IdentityServices
1687 0x00065497 0x100065497 37   38   4.__TEXT.__cstring       ascii   /Users/XpathX/Library/Caches/CloudKit
1688 0x000654bd 0x1000654bd 43   44   4.__TEXT.__cstring       ascii   /Users/XpathX/Library/Containers/com.apple*
1689 0x000654e9 0x1000654e9 53   54   4.__TEXT.__cstring       ascii   /Users/XpathX/Library/Application Support/AddressBook
1690 0x0006551f 0x10006551f 55   56   4.__TEXT.__cstring       ascii   /Users/XpathX/Library/Application Support/CallHistoryDB
1691 0x00065557 0x100065557 42   43   4.__TEXT.__cstring       ascii   /Users/XpathX/Library/Autosave Information
1692 0x00065582 0x100065582 31   32   4.__TEXT.__cstring       ascii   /Users/XpathX/Library/Calendars
1693 0x000655a2 0x1000655a2 30   31   4.__TEXT.__cstring       ascii   /Users/XpathX/Library/Messages
1694 0x000655c1 0x1000655c1 29   30   4.__TEXT.__cstring       ascii   /Users/XpathX/Library/HomeKit
1695 0x000655df 0x1000655df 29   30   4.__TEXT.__cstring       ascii   /Users/XpathX/Library/Sharing
1696 0x000655fd 0x1000655fd 26   27   4.__TEXT.__cstring       ascii   /Users/XpathX/Library/Mail
1697 0x00065618 0x100065618 30   31   4.__TEXT.__cstring       ascii   /Users/XpathX/Library/Accounts
1698 0x00065637 0x100065637 28   29   4.__TEXT.__cstring       ascii   /Users/XpathX/Library/Safari
1699 0x00065654 0x100065654 33   34   4.__TEXT.__cstring       ascii   /Users/XpathX/Library/Suggestions
1700 0x00065676 0x100065676 45   46   4.__TEXT.__cstring       ascii   /Users/XpathX/Library/PersonalizationPortrait
1701 0x000656a4 0x1000656a4 44   45   4.__TEXT.__cstring       ascii   /Users/XpathX/Library/Metadata/CoreSpotlight
1702 0x000656d1 0x1000656d1 31   32   4.__TEXT.__cstring       ascii   /Users/XpathX/Library/Reminders
1703 0x000656f1 0x1000656f1 29   30   4.__TEXT.__cstring       ascii   /Users/XpathX/Library/Cookies
```

Multiple paths for user “XpathX” are embedded in DazzleSpy

There’s no obvious mechanism that would easily result in those being embedded accidentally, and one could be forgiven for thinking that these paths were deliberately placed. We might also wonder about the authenticity of other paths such as `/Users/wangping` /pangu/.

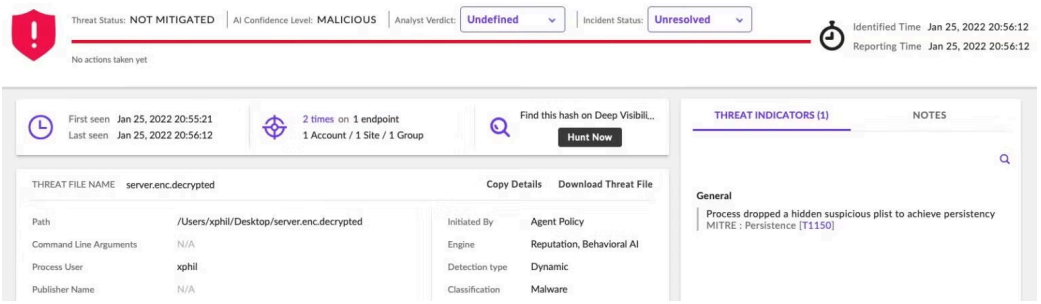
How To Protect Against OSX.DazzleSpy

OSX.DazzleSpy, like macOS.Macma before it, appears to be aimed at visitors to certain websites holding content about, or of interest to, Hong Kong pro-democracy activists and activism. Although that is a small demographic, the threat actors also exploited a (now-patched) local privilege escalation, CVE-2021-30869, to run the payload as root.

SentinelOne’s behavioral engine detects OSX.DazzleSpy on execution. In order to prevent infections like DazzleSpy, be sure to install a good behavioral AI engine that can recognize novel threats based on what they do. Legacy AV scanners that rely on known signatures or cloud reputation services alone will not be able to stop threats that have not previously been detected in the wild.

SentinelOne detects OSX.DazzleSpy on execution

Admin users can view details including threat indicators in the Management console and pivot directly from there to Deep Visibility for extended threat hunting across the estate if required.



The SentinelOne behavioral AI catches the malware attempting persistence

Conclusion

These two new Mac malware families continue trends we noted previously in macOS malware. DazzleSpy's use of vulnerabilities is a clear warning to those that continue to insist Mac users cannot get malware if they engage in "safe behavior": such a stance does not match today's threatscape.

Meanwhile, SysJoker's cross-platform backdoor functionality shows that threat actors are factoring in Mac targets along with Windows and Linux as they develop new ways to steal data and compromise organizations. As with all your other endpoints, it is vital to keep your Mac fleet protected by a capable, defense-in-depth security solution such as the SentinelOne platform.

If you would like to learn more about how SentinelOne can protect your Mac, Windows, Linux, ChromeOS, IoT and Cloud workload endpoints, contact us or request a free demo.

Indicators of Compromise

OSX.SysJoker

DNS REQUESTS
drive.google.com.

googlehosted.l.googleusercontent.com.
graphic-updater.com.

DNS RESPONSES
142.250.199.14
216.58.199.225
216.58.203.78
23.254.131.176
36.4.104.0

COMMANDS EXECUTED
/bin/sh
/bin/bash
/usr/bin/whoami

FILEPATHS
/Users/root/Library/SystemNetwork
~/Library/MacOsServices/updateMacOs

HASHES
updateMacOs
554aef8bf44e7fa941e1190e41c8770e90f07254 1a9a5c797777f37463b44de2b49a7f95abca786db3977dcdac0f79da739c08ac

types-config.ts
01d06375cf4042f4e36467078530c776a28cec05
d0febda3a3d2d68b0374c26784198dc4309dbe4a8978e44bb7584fd832c325f0

OSX.DazzleSpy

FILEPATHS
~/Library/LaunchAgents/com.apple.softwareupdate.plist
~/.local/softwareupdate
~/.local/security.zip
~/.local/security/keystealDaemon
.Documenty/security/libkeystealClient.dylib
.Documenty/security/keys.err
.Documenty/security/security-unsigned
.Documenty/security/keystealDaemon

C2
88.218.192[.]128:5633

HASHES
server.enc
ee0678e58868ebd6603cc2e06a134680d2012c1b
f9ad42a9bd9ade188e997845cae1b0587bf496a35c3bffacd20fefe07860a348
