OceanLotus: macOS malware update

welivesecurity.com/2019/04/09/oceanlotus-macos-malware-update/

April 9, 2019

Latest ESET research describes the inner workings of a recently found addition to OceanLotus's toolset for targeting Mac users

Romain Dumont
9 Apr 2019 - 11:30AM

Latest ESET research describes the inner workings of a recently found addition to OceanLotus's toolset for targeting Mac users

Early in March 2019, a new macOS malware sample from the OceanLotus group was uploaded to VirusTotal, a popular online multi-scanner service. This backdoor executable bears the same features as the previous macOS variant we looked at, but its structure has changed and its detection was made harder. Unfortunately, we couldn't find the dropper associated with this sample so we do not know the initial compromise vector.

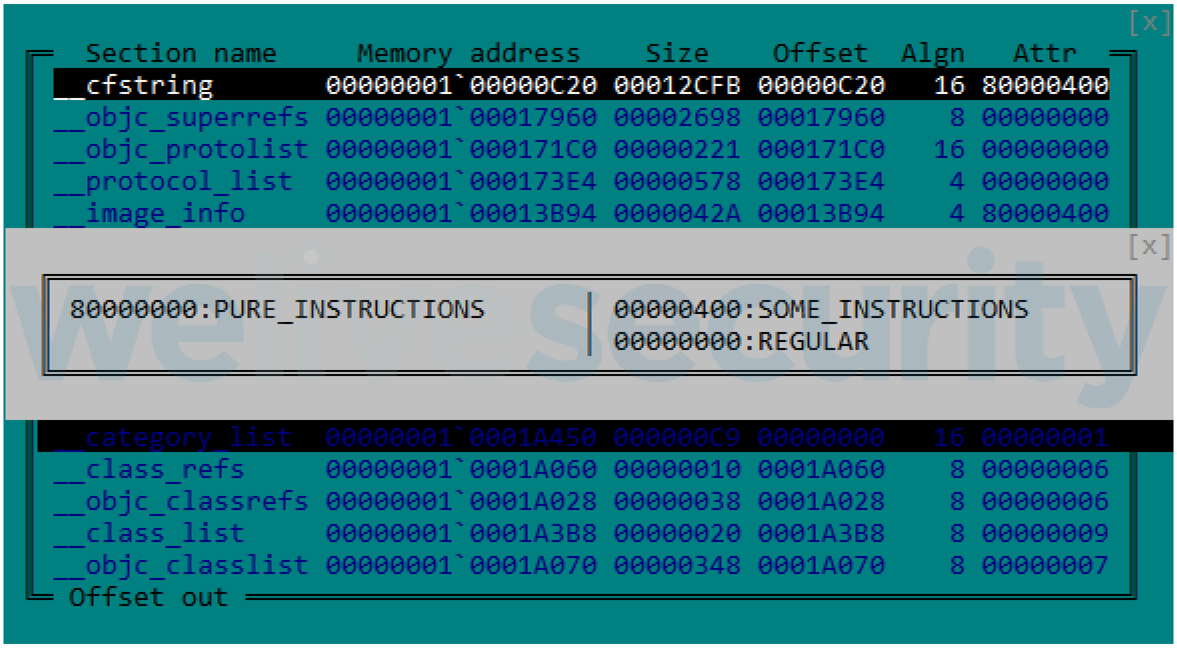
We recently published a detailed update about OceanLotus and how its operators employ a wide range of techniques to gain code execution, achieve persistence, and leave as little trace as possible on a Windows system. OceanLotus is also known to have a malicious macOS component. This article details what has changed from the previous macOS version analyzed by Trend Micro and describes how, while analyzing this variant's code, you can automate string decryption using the IDA Hex-Rays API.

Analysis

The following three sections of this blogpost describe the analysis of the sample with the SHA-1 hash E615632C9998E4D3E5ACD8851864ED09B02C77D2. The file is named flashlightd and is detected by ESET products as OSX/OceanLotus.D.

Anti-debug and anti-sandbox

As usual for OceanLotus macOS binaries, the sample is packed with UPX, but most packer identification tools do not recognize it as such, probably because they mostly include a signature that relies on the presence of a "UPX" string, and further, Mach-O signatures are less common and not as regularly updated. This particular characteristic makes static detection more difficult. Once unpacked, one interesting thing is that the entry point is located at the beginning of the __cfstring section in the .TEXT segment. This section has the flag attributes seen in Figure 1.



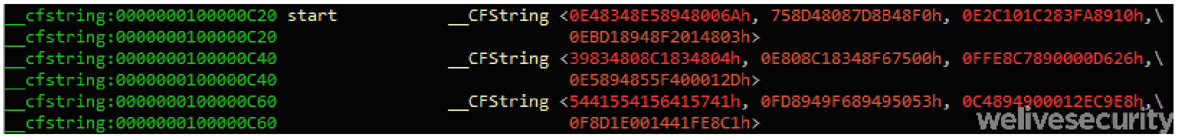
```
  Section name      Memory address     Size     Offset  Algn   Attr
__cfstring        00000001`00000C20 00012CFB 00000C20    16 80000400
__objc_superrefs  00000001`00017960 00002698 00017960     8 00000000
__objc_protolist  00000001`000171C0 00000221 000171C0    16 00000000
__protocol_list   00000001`000173E4 00000578 000173E4     4 00000000
__image_info      00000001`00013B94 0000042A 00013B94     4 80000400

 80000000:PURE_INSTRUCTIONS          00000400:SOME_INSTRUCTIONS
                                     00000000:REGULAR

__category_list   00000001`0001A450 000000C9 00000000    16 00000001
__class_refs      00000001`0001A060 00000010 0001A060     8 00000006
__objc_classrefs  00000001`0001A028 00000038 0001A028     8 00000006
__class_list      00000001`0001A3B8 00000020 0001A3B8     8 00000009
__objc_classlist  00000001`0001A070 00000348 0001A070     8 00000007
 Offset out
```

Figure 1. MACH-O __cfstring section attributes

As seen in Figure 2, the fact that the code is in the __cfstring section tricks some disassembly tools to display the code as strings.

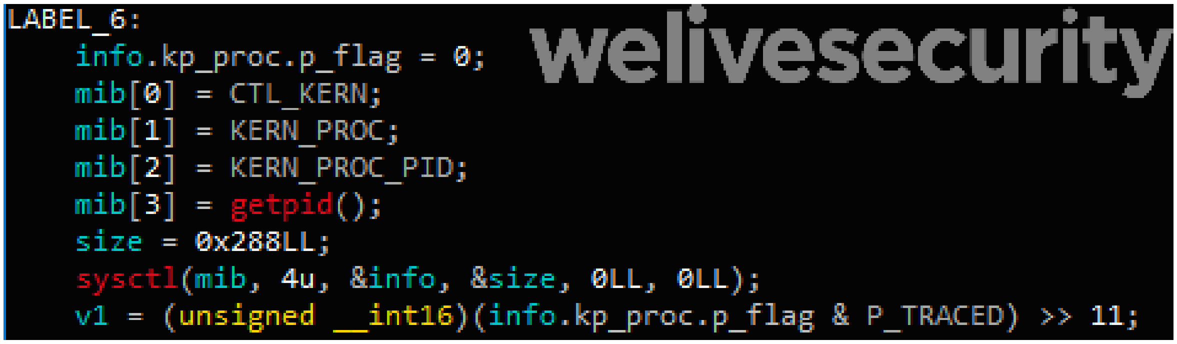


```
__cfstring:0000000100000C20 start            __CFString <0E48348E58948006Ah, 758D48087D8B48F0h, 0E2C101C283FA8910h,\
__cfstring:0000000100000C20                              0EBD18948F2014803h>
__cfstring:0000000100000C40                  __CFString <39834808C1834804h, 0E808C18348F67500h, 0FFE8C7890000D626h,\
__cfstring:0000000100000C40                              0E5894855F400012Dh>
__cfstring:0000000100000C60                  __CFString <5441554156415741h, 0FD8949F689495053h, 0C4894900012EC9E8h,\
__cfstring:0000000100000C60                              0F8D1E001441FE8C1h>
```

Figure 2. The backdoor code is defined as data by IDA

When run, the binary first creates a thread as an anti-debugging watchdog whose sole purpose is to continuously check if a debugger is present. In order to do that, this thread:

- Tries to detach any debugger by calling ptrace with PT_DENY_ATTACH as a request parameter
- Checks if some exception ports are open by calling the task_get_exception_ports function
- Checks if a debugger is attached, as seen in Figure 3, by verifying if the P_TRACED flag is set in the current process



```
LABEL_6:
    info.kp_proc.p_flag = 0;
    mib[0] = CTL_KERN;
    mib[1] = KERN_PROC;
    mib[2] = KERN_PROC_PID;
    mib[3] = getpid();
    size = 0x288LL;
    sysctl(mib, 4u, &info, &size, 0LL, 0LL);
    v1 = (unsigned __int16)(info.kp_proc.p_flag & P_TRACED) >> 11;
```

Figure 3. Check if a debugger is attached via sysctl function

If the watchdog detects that a debugger is present the exit function is called. Moreover, the sample then checks its environment by issuing the following two commands:
ioreg -l | grep -e “Manufacturer” and sysctl hw.model
and checks the return value against a hardcoded list of known virtualization system strings: oracle, vmware, virtualbox or parallels. Finally, the command:
system_profiler SPHardwareDataType 2>/dev/null | awk ‘/Boot ROM Version/ {split($0, line, “:”);printf(“%s”, line[2]);}
checks if the machine is one of the following: “MBP”, “MBA”, “MB”, “MM”, “IM”, “MP” and “XS”. These codes represent the model of the system. For instance, “MBP” stands for MacBook Pro, “MBA” stands for MacBook Air and so on…

Major updates

Even though the backdoor commands have not changed since the Trend Micro article, we noticed a few other modifications. The C&C servers used for this sample are quite recent as their creation date is 2018-10-22.

- daff.faybilodeau[.]com
- sarc.onteagleroad[.]com
- au.charlineopkesston[.]com

The URL resource used has changed to /dp/B074WC4NHW/ref=gbps_img_m-9_62c3_750e6b35.

The first packet that is sent to the C&C server contains more information regarding the host machine. All data gathered by the commands in the following table are included.

Commands	Description
system_profiler SPHardwareDataType 2>/dev/null | awk '/Processor / {split($0,line,":"); printf("%s",line[2]);}' machdep.cpu.brand_string	Gather processor information
system_profiler SPHardwareDataType 2>/dev/null | awk '/Memory/ {split($0,line, ":"); printf("%s", line[2]);}'	Gather memory information
ifconfig -l	Gather network interface MAC addresses
ioreg -rd1 -c IOPlatformExpertDevice | awk '/IOPlatformSerialNumber/ { split($0, line, "\""); printf("%s", line[4]); }'	Retrieves the serial number of the device

On top of this configuration change, this sample does not use the libcurl library for network exfiltration. Instead, it uses an external library. To locate it, the backdoor tries to decrypt each file in the current directory using AES-256-CBC with the key gFjMXBgyXWULmVVVzyxy padded with zeroes. Each file is “decrypted” and saved as /tmp/store and an attempt to load it as a library made using the dlopen function. When a decryption attempt results in a successful call to dlopen, the backdoor then retrieves the exported functions Boriry and ChadylonV, which seem to be responsible for the network communication with the server. As we do not have the dropper or other files from the original sample’s location, we could not analyse this library. Moreover, since the component is encrypted, a YARA rule based on these strings would not match the file found on disk.

As described in the analysis of the group’s previous macOS backdoor, a *clientID* is created. This identifier is the MD5 hash of the return value of one of the following commands:

- ioreg -rd1 -c IOPlatformExpertDevice | awk ‘/IOPlatformSerialNumber/ { split($0, line, “\””); printf(“%s”, line[4]); }’

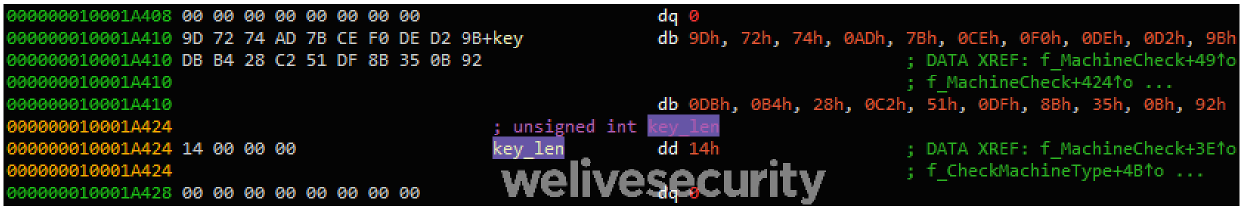
- ioreg -rd1 -c IOPlatformExpertDevice | awk '/IOPlatformUUID/ { split($0, line, "\""); printf("%s", line[4]); }'
- ifconfig en0 | awk \'/ether /{print $2}\' (obtain the MAC address)
- an unknown command ("\x1e\x72\x0a") which used to be "uuidgen" in the previous samples

Before being hashed, the character "0" or "1" is appended to the return value indicating root privileges. This *clientID* is stored in /Library/Storage/File System/HFS/25cf5d02-e50b-4288-870a-528d56c3cf6e/pivtoken.appex if the code runs as root, or in ~/Library/SmartCardsServices/Technology/PlugIns/drivers/snippets.ecgML otherwise. This file is normally hidden via the _chflags function and its timestamp is modified using the "touch –t" command with a random value.

String decryption

Like previous variants, the strings are encrypted using AES-256-CBC (hex-encoded key: 9D7274AD7BCEF0DED29BDBB428C251DF8B350B92 padded with zeroes and the IV is filled with zeroes) using the CCCrypt function. The key has changed from previous versions but since the group is still using the same algorithm to encrypt strings, decryption could be automated. Along with this article, we are releasing an IDA script leveraging the Hex-Rays API to decrypt the strings present in the binary. This script may help future analysis of OceanLotus and the analysis of existing samples that we have not yet been able to obtain. At the core of this script lies a generic method to obtain the arguments passed to a function. Moreover, it looks for the parameter assignments in order to find their values. This method could be reused to retrieve the list of arguments of a function and then pass them to a callback.

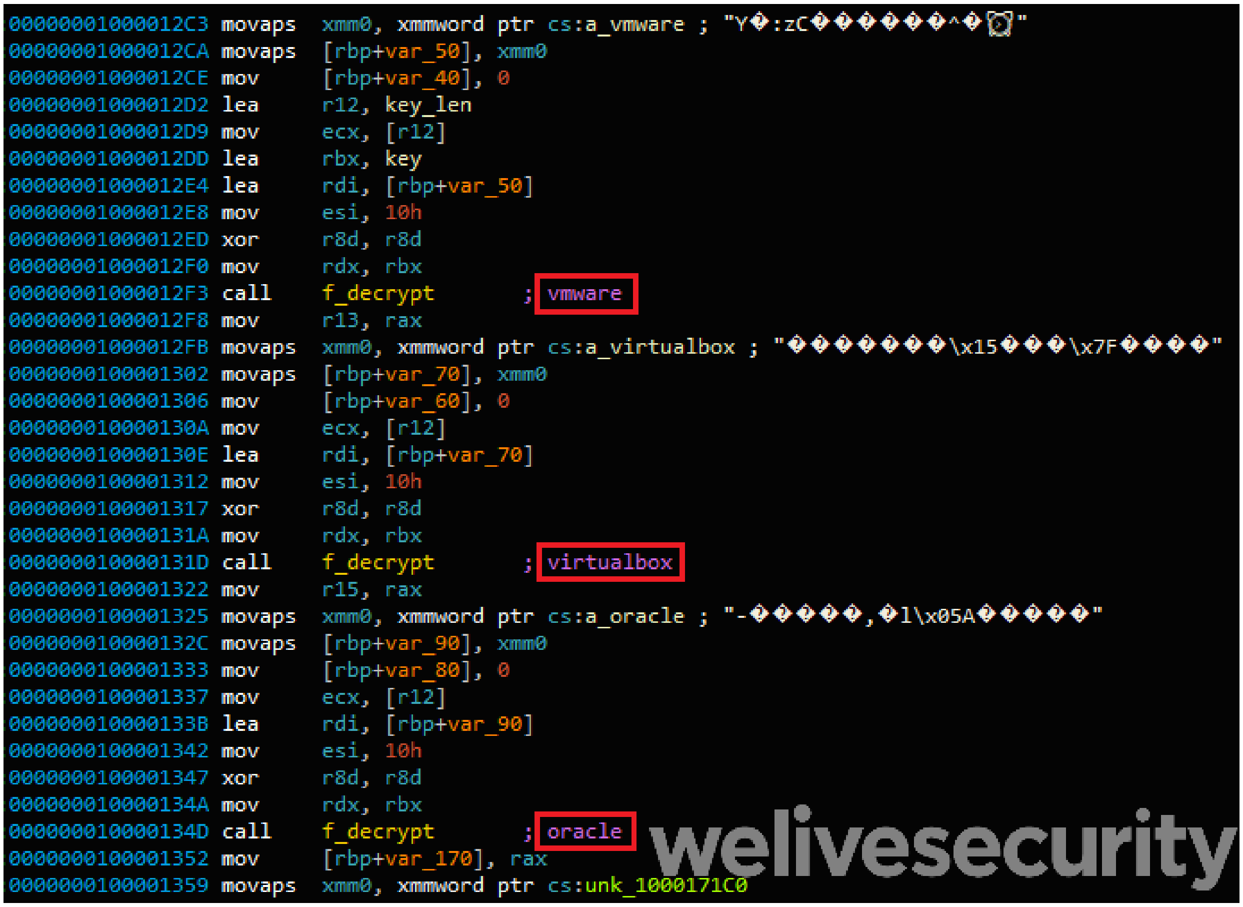
Knowing the prototype of the *decrypt* function, the script first finds all cross-references to this function, finds all the arguments, decrypts the data and puts the plaintext inside a comment at the address of the cross-reference. In order for the script to work correctly, the custom alphabet used by the base64 decode function must be set in the script and the global variable containing the length of the key must be defined (as a DWORD in this case; see Figure 4).



```
000000010001A408 00 00 00 00 00 00 00 00                  dq 0
000000010001A410 9D 72 74 AD 7B CE F0 DE D2 9B+key        db 9Dh, 72h, 74h, 0ADh, 7Bh, 0CEh, 0F0h, 0DEh, 0D2h, 9Bh
000000010001A410 DB B4 28 C2 51 DF 8B 35 0B 92                                            ; DATA XREF: f_MachineCheck+49↑o
000000010001A410                                                                          ; f_MachineCheck+424↑o ...
000000010001A410                                          db 0DBh, 0B4h, 28h, 0C2h, 51h, 0DFh, 8Bh, 35h, 0Bh, 92h
000000010001A424                          ; unsigned int key_len
000000010001A424 14 00 00 00              key_len         dd 14h                          ; DATA XREF: f_MachineCheck+3E↑o
000000010001A424                                                                          ; f_CheckMachineType+4B↑o ...
000000010001A428 00 00 00 00 00 00 00 00                  dq 0
```

Figure 4. Defining the global variable key_len

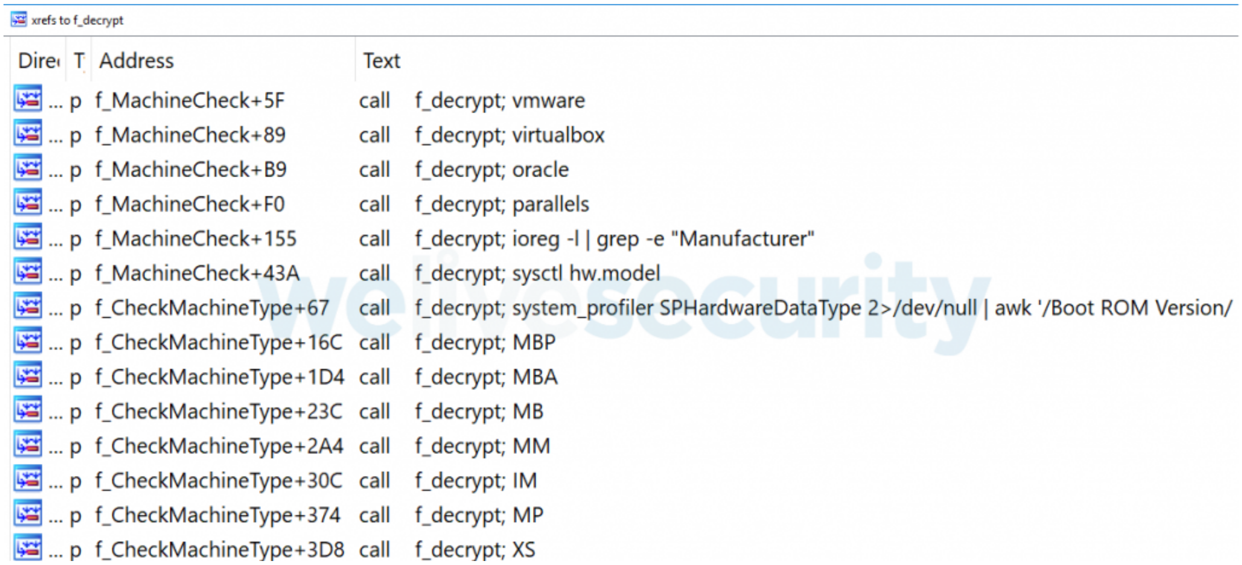
In the *Function* window, you can right-click the decryption function and click "Extract and decrypt arguments". The script should put the decrypted strings in comments, much as in Figure 5.



```
00000001000012C3 movaps  xmm0, xmmword ptr cs:a_vmware ; "Y�:zC������^�⏰"
00000001000012CA movaps  [rbp+var_50], xmm0
00000001000012CE mov     [rbp+var_40], 0
00000001000012D2 lea     r12, key_len
00000001000012D9 mov     ecx, [r12]
00000001000012DD lea     rbx, key
00000001000012E4 lea     rdi, [rbp+var_50]
00000001000012E8 mov     esi, 10h
00000001000012ED xor     r8d, r8d
00000001000012F0 mov     rdx, rbx
00000001000012F3 call    f_decrypt       ; vmware
00000001000012F8 mov     r13, rax
00000001000012FB movaps  xmm0, xmmword ptr cs:a_virtualbox ; "�������\x15���\x7F����"
0000000100001302 movaps  [rbp+var_70], xmm0
0000000100001306 mov     [rbp+var_60], 0
000000010000130A mov     ecx, [r12]
000000010000130E lea     rdi, [rbp+var_70]
0000000100001312 mov     esi, 10h
0000000100001317 xor     r8d, r8d
000000010000131A mov     rdx, rbx
000000010000131D call    f_decrypt       ; virtualbox
0000000100001322 mov     r15, rax
0000000100001325 movaps  xmm0, xmmword ptr cs:a_oracle ; "-�����,�l\x05A�����"
000000010000132C movaps  [rbp+var_90], xmm0
0000000100001333 mov     [rbp+var_80], 0
0000000100001337 mov     ecx, [r12]
000000010000133B lea     rdi, [rbp+var_90]
0000000100001342 mov     esi, 10h
0000000100001347 xor     r8d, r8d
000000010000134A mov     rdx, rbx
000000010000134D call    f_decrypt       ; oracle
0000000100001352 mov     [rbp+var_170], rax
0000000100001359 movaps  xmm0, xmmword ptr cs:unk_1000171C0
```

Figure 5. Decrypted text is put into comments

This conveniently lists the decrypted strings together in IDA's *xrefs to* window for that function, as seen in Figure 6.



Dire	T	Address	Text
...	p	f_MachineCheck+5F	call f_decrypt; vmware
...	p	f_MachineCheck+89	call f_decrypt; virtualbox
...	p	f_MachineCheck+B9	call f_decrypt; oracle
...	p	f_MachineCheck+F0	call f_decrypt; parallels
...	p	f_MachineCheck+155	call f_decrypt; ioreg -l | grep -e "Manufacturer"
...	p	f_MachineCheck+43A	call f_decrypt; sysctl hw.model
...	p	f_CheckMachineType+67	call f_decrypt; system_profiler SPHardwareDataType 2>/dev/null | awk '/Boot ROM Version/
...	p	f_CheckMachineType+16C	call f_decrypt; MBP
...	p	f_CheckMachineType+1D4	call f_decrypt; MBA
...	p	f_CheckMachineType+23C	call f_decrypt; MB
...	p	f_CheckMachineType+2A4	call f_decrypt; MM
...	p	f_CheckMachineType+30C	call f_decrypt; IM
...	p	f_CheckMachineType+374	call f_decrypt; MP
...	p	f_CheckMachineType+3D8	call f_decrypt; XS

Figure 6. Xrefs to of f_decrypt function

The final script can be found on our Github repository.

Conclusion

As recently documented in another of our articles, the OceanLotus group keeps improving and updating its toolset, and once again, it has improved its tools for targeting Mac users. The code has not changed that much, but because many Mac users don’t run security software on their machines, the need to evade detection is of less importance. ESET products already detected this file when we found it. Since the network library used for the C&C communication is now encrypted on the disk, the exact network protocol used remains unknown.

Indicators of Compromise (IoCs)

The IoCs in this blogpost, as well as the MITRE ATT&CK attributes, are also available in our GitHub repository.

Domain names

- daff.faybilodeau[.]com
- sarc.onteagleroad[.]com
- au.charlineopkesston[.]com

URL resource

/dp/B074WC4NHW/ref=gbps_img_m-9_62c3_750e6b35

File paths

- ~/Library/SmartCardsServices/Technology/PlugIns/drivers/snippets.ecgML
- /Library/Storage/File System/HFS/25cf5d02-e50b-4288-870a-528d56c3cf6e/pivtoken.appex
- /tmp/store

Sample analyzed	SHA-1 hash	ESET detection name
fleshlightd	E615632C9998E4D3E5ACD8851864ED09B02C77D2	OSX/OceanLotus.D

MITRE ATT&CK techniques

Tactic	ID	Name	Description
Defense Evasion	**T1158**	Hidden Files and Directories	The backdoor hides the *clientID* file via **chflags** function.

Tactic	ID	Name	Description
	T1107	File Deletion	The backdoor can receive a “delete” command.
	T1222	File Permissions Modification	The backdoor changes the permission of the file it wants to execute to 755.
	T1027	Obfuscated Files or Information	The library used for network exfiltration is encrypted with AES-256 in CBC mode.
	T1099 (macOS)	Timestomp	The timestamp of the file storing the clientID is modified with a random value.
Discovery	T1082	System Information Discovery	The backdoor performs a fingerprint of the machine on its first connection to the C&C server.
Exfiltration	T1022	Data Encrypted	The backdoor encrypts the data before exfiltration.
Command and Control	T1094	Custom Command and Control Protocol	The backdoor implements a specific format for the packet involving random values. See Trend Micro article.

9 Apr 2019 - 11:30AM

Sign up to receive an email update whenever a new article is published in our Ukraine Crisis – Digital Security Resource Center

Newsletter

Discussion