

APT32海莲花组织最新活动： 老技术，新后门

March 23, 2018 • [ang010ela](#)

OceanLotus（海莲花），也称APT32, APT-C-00，攻击目标主要是东亚国家。研究表明该组织一直在持续更新后门、基础设施和感染单元。海莲花攻击的主要目标是东亚国家的企业和政府组织。几个月前，研究人员发现了该组织的最新后门并进行了分析。

传播

攻击者使用不同的方法欺骗潜在的受害者允许恶意dropper。监测发现越南、菲律宾、老挝、柬埔寨是东亚被攻击最多的国家。

双扩展和假应用图标(Word, PDF)

Droppers一般是以邮件附件形式出现的，一些文件名包括：

```
Mi17 Technical issues - Phonesack Grp.exe. Mi-17 是俄罗斯的直升机型号。 is a common Russian helicopter.  
Chi tiet don khieu nai gui saigontel.exe, 该应用将用户对Saigontel的抱怨发送给Saigontel, Saigontel是一家越南电信公司。  
Updated AF MOD contract - Jan 2018.exe  
remove_pw_Reschedule of CISC Regular Meeting.exe  
Sorchornor_with_PM_-_Sep_2017.exe  
20170905-Evaluation Table.xls.exe  
CV_LeHoangThing.doc.exe. 一些假的简历文件也出现在加拿大。
```

这些文件有一个共同点，那就是都是用了密码保护的诱饵文件。而密码提供的方式是不确定的，有时候会出现在邮件中，有时候这些文件本来就不能工作。

假安装包

一些假的安卓包，声称是一些流行软件的安装包或者更新包，这在水坑攻击中是常用的。比如，Freebuf上一篇关于重封装的Firefox安装包。另一个样本是，该样本通过被入侵的网站进行传播，但是目前没有足够的证据。

所有上述的文件，无论是通过邮件还是访问被入侵的网站，都会释放相同的后门组件。随后会对样本 RobototFontUpdate.exe 进行分析，了解在系统中执行恶意 payload 的过程。

技术分析

安装和执行的过程依赖于多层的混淆技术，比如 payload 的解密、PE 重构和加载 shellcode，以及侧加载技术等。

执行流概览

攻击一共分为二个部分，dropper 和后门加载器。下面是恶意软件执行的流图。

Dropper部分的执行流如下：

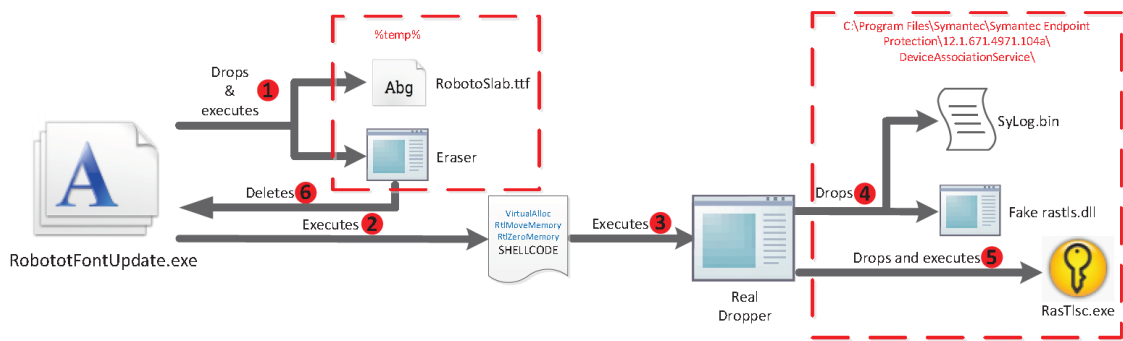


Figure 1 Dropper execution flow

图 1 Dropper执行流

后门部分执行流如下：

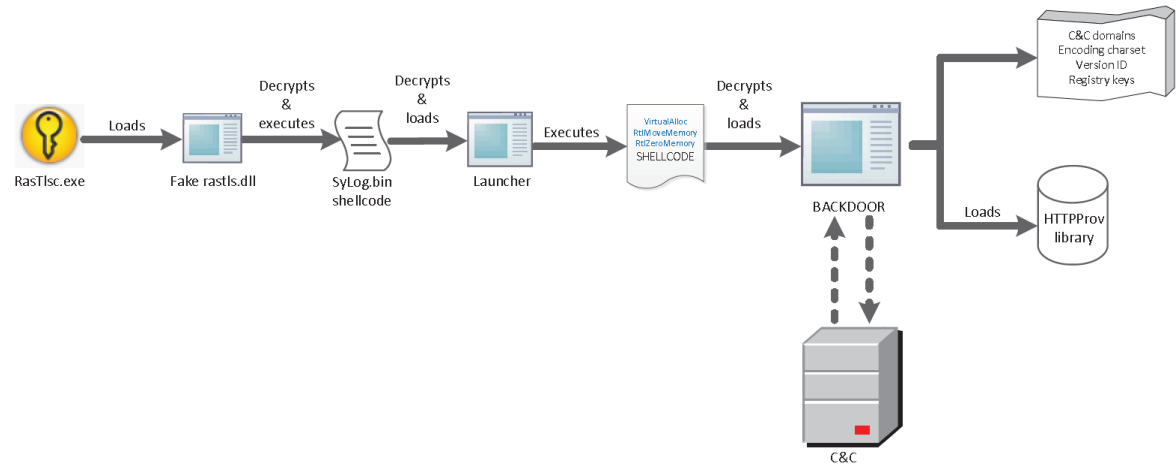


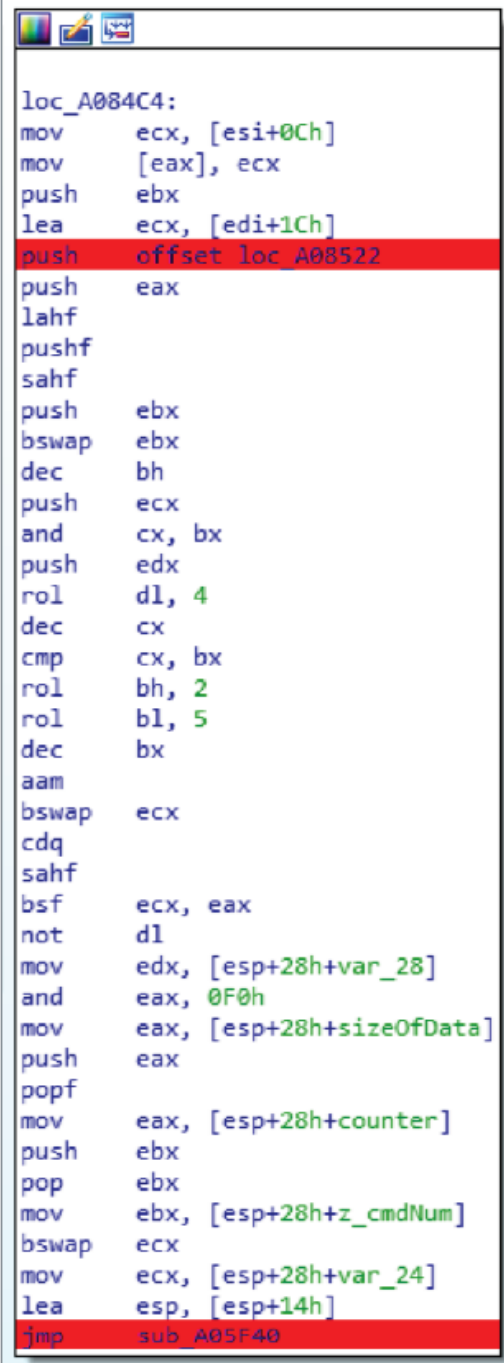
Figure 2 Backdoor execution flow

控制流混淆

几乎所有的组件都被混淆了，混淆是基于补充的条件跳转指令对进行的，使用的形式有JZ/JNZ, JP/JNP, JO/JNO等；而且每对都跳转到同一目标。该序列是和垃圾代码交错出现的，其中垃圾代码使用了栈指针，但是并不改变条件flag的值。也就是说会在同一分支结束执行。因为使用了正的栈指针值，所以在反编译时会引发一些问题。



一些基本块会向栈中push一个地址，以JMP/CALL结束；而其他基本块会push两个地址以RET指令结束。第二个push是调用函数，第一个push是要挑战的下一块地址。这就会创建一些没有父母的基本块。



```
loc_A084C4:
mov     ecx, [esi+0Ch]
mov     [eax], ecx
push    ebx
lea     ecx, [edi+1Ch]
push    offset loc_A08522
push    eax
lahf
pushf
sahf
push    ebx
bswap   ebx
dec     bh
push    ecx
and     cx, bx
push    edx
rol     dl, 4
dec     cx
cmp     cx, bx
rol     bh, 2
rol     bl, 5
dec     bx
aam
bswap   ecx
cdq
sahf
bsf     ecx, eax
not     dl
mov     edx, [esp+28h+var_28]
and     eax, 0F0h
mov     eax, [esp+28h+sizeOfData]
push    eax
popf
mov     eax, [esp+28h+counter]
push    ebx
pop     ebx
mov     ebx, [esp+28h+z_cmdNum]
bswap   ecx
mov     ecx, [esp+28h+var_24]
lea     esp, [esp+14h]
jmp     sub_A05F40
```

Figure 4 PUSH/JMP technique

在了解了方案后，很容易就可以标记垃圾代码，在分析代码时就可以忽略这些垃圾代码。

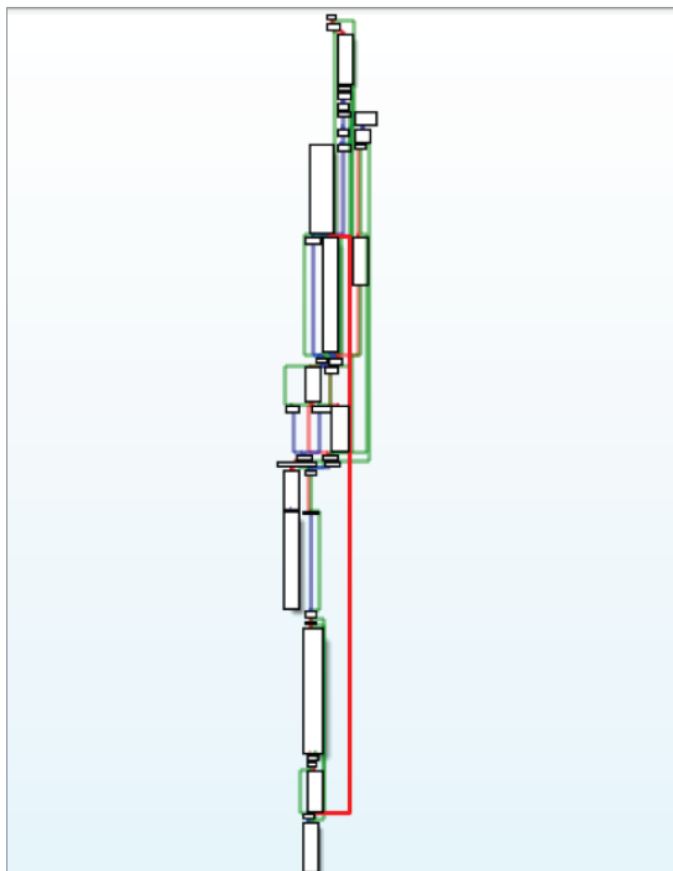


Figure 5 **Control flow obfuscation**

Dropper

Stage1 诱饵文件



Figure 6 **RobototFontUpdate icon**

在过去几个月里，使用的诱饵文件有很多，其中包括Roboto Slab字体的假的TrueType更新。选择该字体有点奇怪，因为该字体支持的东亚语言并不多。执行后，二进制文件会解密源并解压缩解密的数据（LZMA）。合法的RobotoSlab-Regular.ttf(SHA1:912895e6bb9e05af3a1e58a1da417e992a71a324)文件会写入%temp%文件夹，并通过win32 API函数ShellExecute运行。

从源中解密的shellcode就执行了，执行后，假的字体更新会释放另一个应用，该应用的目的是删掉dropper。这个擦除的应用释放在%temp%\[0-9].tmp.exe下。

Stage 2 shellcode

每个阶段使用的shellcode都是相同的。这里的shellcode是一个传统的PE加载器，会在内存中创建可执行文件，解密所有的部分并计算必要的重新排列和其他偏移。Shellcode会提取3个Windows API函数：VirtualAlloc, RtlMoveMemory 和RtlZeroMemory。

RtlZeroMemory 函数在PE头中常用的zero-out域。因为MZ/PE头被破坏了，所以依赖自动化内存复制是没用的。Shellcode会调用解密的PE中的入口函数，然后调用DLLEntry出口函数。

Stage 3 Real Dropper, {103004A5-829C-418E-ACE9-A7615D30E125}.dll

该可执行文件会通过Windows API用AES算法的CBC模式解密源。硬编码的key大小是256位。解密后，数据会被解压（LZMA算法）。

如果该进程是以管理员权限运行，恶意软件会创建一个服务或者用经典的Windows “Run”注册表 (HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run;DeviceAssociationService;rastlsc.exe)来实现驻留。

如果 dropper 是以管理员权限运行，那么就会尝试在 C:\Program Files\Symantec\Symantec Endpoint Protection\12.1.671.4971.104a\DeviceAssociationService\文件夹下写入一些文件，或者写入%APPDATA%\Symantec\Symantec

- rastlsc.exe (SHA1:2616da1697f7c764ee7fb558887a6a3279861fac, 合法的symantec网络访问控制应用dot1xtra.exe的副本)
- SyLog.bin (SHA1:5689448b4b6260ec9c35f129df8b8f2622c66a45, 加密后门)
- rastls.dll (SHA1:82e579bd49d69845133c9aa8585f8bd26736437b,rastlsc.exe侧加载的恶意DLL)

不同样本的路径是不同的，但是模式是类似的。

基于权限，恶意软件会在%ProgramFiles%或%appdata%中释放文件。

- \Symantec\CNG Key Isolation\
- \Symantec\Connected User Experiences and Telemetry\
- \Symantec\DevQuery Background Discovery Broker Tasks\

这些路径是不同的Symantec产品使用的。

在完成驻留和释放可执行文件后，合法的Symantec rastlsc.exe会用CreateProcessW执行，另外一个版本是{BB7BDEC9-B59D-492E-A4AF-4C7B1C9E646B}.dll，在该版本中，rastlsc.exe会以参数krv运行。

后门组件： rastlsc.exe side-loading

海莲花组织在Symantec产品的可执行文件中使用了一种古老但是公开的技术，通过在系统文件夹中写入恶意库来利用合法签名的可执行文件的库加载过程。这样，就会让恶意行为看起来是合法的，因为这些行为是可信的可执行进程执行的。合法的可执行文件rastlsc.exe就会释放和执行。

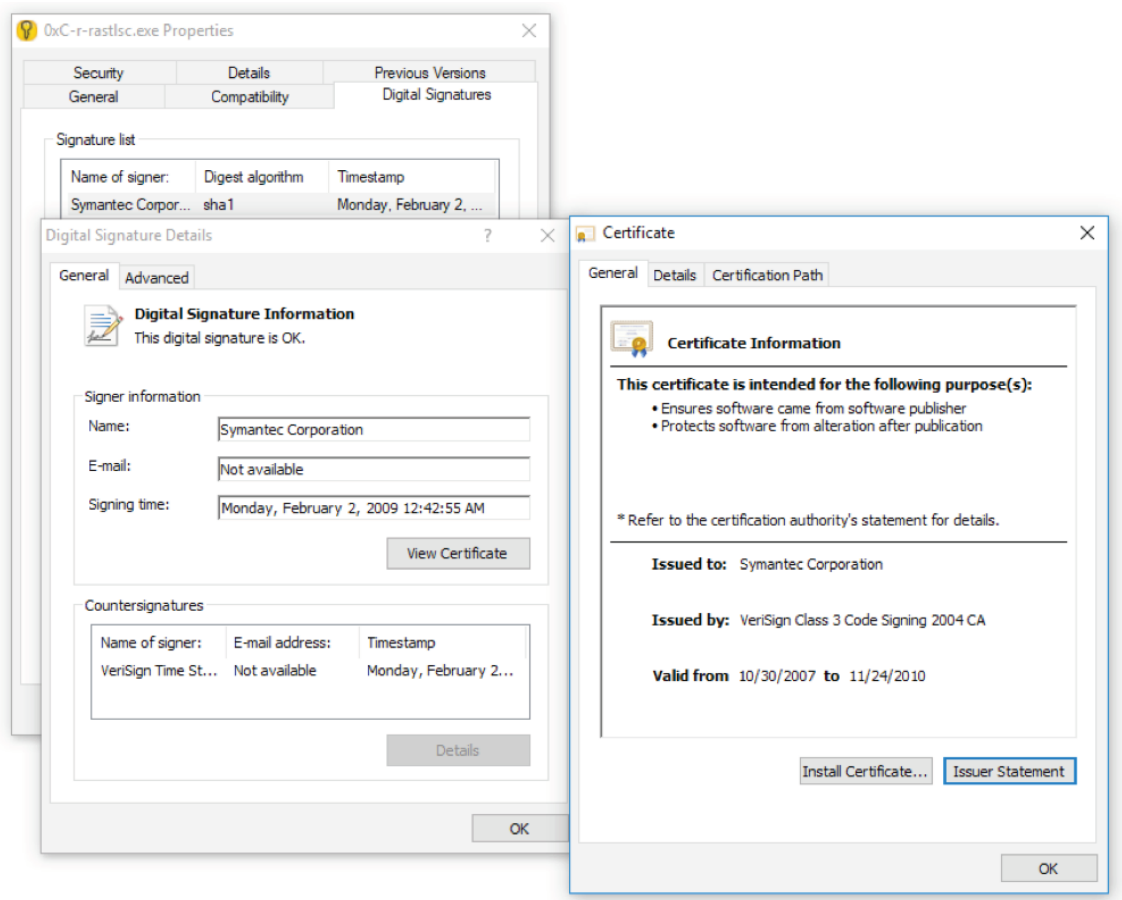
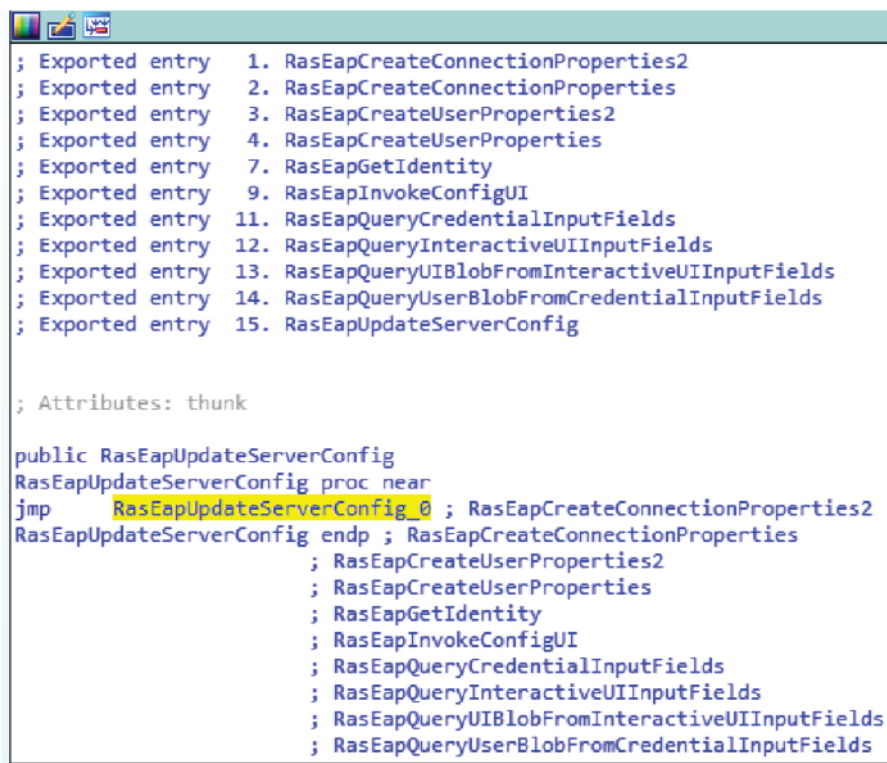


Figure 7 Symantec rastlsc.exe digital signature

可执行文件会输入rastls.dll文件，该文件中含有恶意payload。

侧加载也可以用其他合法签名的可执行文件包括McAfee的mcoemcpy.exe，该文件会加载McUtil.dll。这种技术之前被PlugX使用过，也受到了越南CERT的关注。

Stage 1 库函数侧加载rastls.dll



```
; Exported entry 1. RasEapCreateConnectionProperties2
; Exported entry 2. RasEapCreateConnectionProperties
; Exported entry 3. RasEapCreateUserProperties2
; Exported entry 4. RasEapCreateUserProperties
; Exported entry 7. RasEapGetIdentity
; Exported entry 9. RasEapInvokeConfigUI
; Exported entry 11. RasEapQueryCredentialInputFields
; Exported entry 12. RasEapQueryInteractiveUIInputFields
; Exported entry 13. RasEapQueryUIBlobFromInteractiveUIInputFields
; Exported entry 14. RasEapQueryUserBlobFromCredentialInputFields
; Exported entry 15. RasEapUpdateServerConfig

; Attributes: thunk

public RasEapUpdateServerConfig
RasEapUpdateServerConfig proc near
jmp RasEapUpdateServerConfig_0 ; RasEapCreateConnectionProperties2
RasEapUpdateServerConfig endp ; RasEapCreateConnectionProperties
; RasEapCreateUserProperties2
; RasEapCreateUserProperties
; RasEapGetIdentity
; RasEapInvokeConfigUI
; RasEapQueryCredentialInputFields
; RasEapQueryInteractiveUIInputFields
; RasEapQueryUIBlobFromInteractiveUIInputFields
; RasEapQueryUserBlobFromCredentialInputFields
```

Figure 8 All rasltls.dll exports lead to the same function

Dll的内部文件名为{7032F494-0562-4422-9C39-14230E095C52}.dll，但是研究人员还看到{5248F13C-85F0-42DF-860D-1723EEAA4F90}.dll这样的版本，所有的输出函数都会指向同一函数的执行。

输出函数会尝试读取位于同一文件夹下的SyLog.bin文件，其中版本会尝试打开文件OUTLFLTR.DAT。如果该文件存在，就会使用AES在CBC模式下解密，然后解压缩（LZMA压缩）。

McUtil.dll变种会使用一种不同的技术。看起来主函数没有任何恶意行为，但事实上会替换合法mcoemcpy.exe文件的.text部分，该文件是一个侧加载的二进制文件。会生成调用读取mcscentr.adf文件中加密的stage-two shellcode的函数的shellcode。

下图是创建shellcode的伪代码：


```

x = False
i = 0
buff = genRandom()
opc1 = [0x58, 0x59, 0x5a, 0x5b]
opc2 = [0x50, 0x51, 0x52, 0x53, 0x54, 0x55, 0x56, 0x57]
opc3 = [0x90, 0x40, 0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48,
        0x49, 0x4a, 0x4b]
while i < len(buff):
    currentChar = buff[i]
    if currentChar < 0xc8:
        buff[i] = opc1[currentChar % len(opc1)]
    else:
        if x:
            buff[i] = opc2[currentChar % len(opc2)]
        else:
            buff[i] = opc3[currentChar % len(opc3)]
        x = x == False
    i+=1

```

结果就是下面的集合列表：

90	nop
4A	dec edx
43	inc ebx
4A	dec edx
90	nop
48	dec eax
48	dec eax
48	dec eax
48	dec ebx
41	inc ecx
49	dec ecx
90	nop
40	inc eax
90	nop
48	dec ebx
48	dec ebx
4A	dec edx
49	dec ecx
53	push ebx
41	inc ecx
B8 D0 1E B6 73	mov eax,0xb-r-mcutil.73B61ED0
FF D0	call eax
C3	ret

Figure 9 Generated shellcode

Stage 2~Stage 4 Shellcode, Launcher, Shellcode

Shellcode会解密并加载库{E1E4CBED-5690-4749-819D-24FB660DF55F}.dll。提取源并尝试开启DeviceAssociationService服务。解密的数据也含有shellcode，后者会解密最后的一层，也就是后门。

变种 {92BA1818-0119-4F79-874E-E3BF79C355B8}.dll会检查rastlsc.exe有没有用krv作为第一参数执行。如果krv以第一参数执行，就会创建一个任务，然后rastlsc.exe文件也会在没有参数的情况下再执

行一次。

Stage 5 后门 {A96B020F-0000-466F-A96D-A91BBF8EAC96}.dll

恶意软件首先尝试从源中提取并用RC4解密。解密的源含有一些用来配置后门的信息。该配置的格式可以直接逆向。用Kaitai struct和structure dumper可以看到：

```
[.] [root]
[.] total_length = 345522
[-] data_n (1 = 0x1 entries)
[-] 0
[.] reg_part_len = 298
[-] domain_encoding_str
[.] str_len = 20
[.] str_buf = "ghijklmnop"
[-] first_reg
[.] str_len = 122
[.] str_buf = "SOFTWARE\\App\\AppX3bbba44c6cae4d9695755183472171e2\\Application"
[-] second_reg
[.] str_len = 122
[.] str_buf = "SOFTWARE\\App\\AppX3bbba44c6cae4d9695755183472171e2\\DefaultIcon"
[-] reg_value
[.] str_len = 8
[.] str_buf = "Data"
[-] mutex_encoding_str
[.] str_len = 6
[.] str_buf = "vwx"
[.] domain_part_len = 100
[-] domains_str (3 = 0x3 entries)
[-] 0
[.] str_len = 28
[.] str_buf = "traveroyce.com"
[-] 1
[.] str_len = 36
[.] str_buf = "braydenhateaub.com"
[-] 2
[.] str_len = 24
[.] str_buf = "antennham.com"
[.] httpprov_len = 345096
[.] httpprov = 00 00 00 00 00 44 05 00 4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00 b8 00
[.] backdoor_version_len = 4
[.] backdoor_version = 24 75 b4 09
```

Figure10 Configuration structure

图 10 配置结构

NOTES: 除了字符串domain_encoding_str和 httpprov库外，其他的数据在每个变种中都会变化。注册表几乎是一样的，因为遵循系统的模式：

```
\HKCU\SOFTWARE\Classes\AppX[a-f0-9]{32}
```

恶意软件会提取用户名的前10个字节，用UTF-16 mutex_encoding_str字符串的3个字母进行XOR运算，然后用16进制编码。计算的结果作为一个互斥名。比如，用户名以abc开头，key是vwx，那么mutex就是：

```
\Sessions\1\BaseNamedObjects\170015001b
```

后门中还有一个PE加载器文件可以在内存中加载HTTPProv.dll库函数，称之为入口点，然后调用输出函数CreateInstance。

通信

后门使用了经典的TCP通讯协议，使用端口25123。为了获取服务器的IP地址，后门首先会创建一个特殊的DNS请求。恶意软件会在配置中的3个域名中选择一个，添加用2个值生成的传统子域名。第一个值是16个字节长的计算机名，第二个值是4字节的版本ID。后面的Python 2代码实现编码算法：

```
letters=domain_encoding_str # "ghijklmnop"
hex_pc_name=pc_name.encode("UTF-16LE").encode("hex")
s=''
for c in hex_pc_name:
    if 0x2f < ord(c) < 0x3a:
        s+=letters[ord(c) - 0x30]
    else:
        s+=c
```

如果计算机名是随机的，版本ID是0x0a841523，就会创建这样的域名：

```
niggmhggmeggmkggmfggmdggidggngggmjgg.ijhlokga.dwarduong[.]com
```

下面的表达可以用来标记C&C服务器：

```
[ghijklmnopabcdef]{4-60}\.[ghijklmnopabcdef]{8}\.[a-z]+\.[a-z]+
```

如过从一个特定的域名中去解析IP地址，恶意软件会在25123端口上建立TCP连接。每个表有3个不同的域名可以用来寻找C&C服务器。所有的通信都是用RC4加密的，并用LZMA压缩。因为key是预先加在包中的，所以也可能解密流量。格式是：

```
[RC4 key (4 bytes)][encrypted data]
```

Key的每个字节都是用随机函数生成的。一旦包被解密和解压缩，数据格式如下：

```
[dw:unknown][dw:unknown][dw:command number][dw:size of data][dw:unknown]
[dw:data]
```

客户端第一次连接服务器，服务器会返回一个UUID作为session ID。Session ID会以二进制数据的形式保存在注册表中：

```
HKCU\SOFTWARE\Classes\AppXc52346ec40fb4061ad96be0e6cb7d16a\DefaultIcon
```

后门中还有一个叫做HTTPprov的库，库是一种与服务器通信的可选方式，也就是备份。DLL会发送post请求与之通信，同时支持HTTPS和SOCKS5,SOCKS4a,SOCKS4的使用。该库是与libcurl静态链接的。

一旦初始化完成，就会创建之后的注册表来指引后门使用HTTP和C&C服务器通信：

```
HKCU\SOFTWARE\Classes\CLSID{E3517E26-8E93-458D-A6DF-8030BC80528B}.
```

通用的用户代理是Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.0;Trident/4.0).

库中最特别的特征是传统的URI编码算的。URI的资源部分是用下面的伪代码创建的：

```
buffEnd = ((DWORD)genRand(4) % 20) + 10 + buff;
while (buff < buffEnd){
    b=genRand(16);
    if (b[0] - 0x50 > 0x50)
        t=0;
    else
        *buf++= UPPER(vowels[b[1] % 5]);
    v=consonants[b[1]%21]);
    if (!t)
        v=UPPER(v);
    *buff++= v;
    if (v!='h' && b[2] - 0x50 < 0x50)
        *buff++= 'h';
    *buff++= vowels[b[4] % 5];
    if (b[5] < 0x60)
        *buff++= vowels[b[6] % 5];
    *buff++= consonants[b[7] % 21];
    if (b[8] < 0x50)
        *buff++= vowels[b[9] % 5];
    *buff++= '-';
};
*buff='\0';
```

从生成的字符串中，我们可以看出，基于传统的校验和计算两个数来获取URI:

```
checksum=crc32(buff)
num2=(checksum >> 16) + (checksum & 0xffff) * 2
num1=(num2 ^ 1) & 0xf
URL=GENERATED_DOMAIN+ "/" + num1 + "/" + num2 + "-" + buff
```

通过增加HTTPprov库的URI生成器，可以生成下面的URL:

```
hXXp://niggmhggmeggmkggmfggmddgidgngggmjgg.ijhlokga.aisicoin[.]com/
13/139756-Ses-Ufali-L
```

命令

在获取sessionID后，后门会做一个系统的指纹。包也是照这样做的：

Offset in the packet	Description
0x000	byte: value varies in each version
0x001	0x01: hardcoded byte
0x002	bool: is elevation token present
0x003	dword: version ID
0x007	string (UTF-16), computer name (0x20 bytes max)
0x027	string (UTF-16), user name
0x079	registry query result of the HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion values: ProductName, CSDVersion, CurrentVersion, ReleaseId, CurrentBuildNumber and the result of the call to IsWow64Process (x86 x64)
0x179	Following format string "%s(%s)," replaced with (GetVolumeInformationW: VolumeNameBuffer), VolumePathNames
0x279	PhysicalDrive deviceIOControl 0x2D1400 (IOCTL_STORAGE_QUERY_PROPERTY) (VolSerialNumber)
0x379	wmi SELECT SerialNumber FROM Win32_BaseBoard
0x3f9	GetSystemTimeAsFileTime
0x400	bool: unknown
0x401	dword: obtained from the decryption of the resource

下面是一个系统指纹的例子：

```
Fingerprint
=====
0x0:03
0x1:01
0x2:  isAdmin ? : 01
0x3:  Version: 0xa841523
0x7:  ComputerName: MOISEPC
0x27: Username: Administrator
0x79: RegistryQueries: Windows 7 Enterprise Service Pack 1 6.1.7601 x86
0x179: Volumes: ( C : ) ; ( D : ) ;

0x279: VolumeSerialNumber: [REDACTED] - [REDACTED] - [REDACTED] - [REDACTED] - [REDACTED]

0x379: BaseBoard Serial Number: [REDACTED]
0x3f9: SystemTimeAsFileTime: 0x1BF545D479FA600
```

Figure 11 System fingerprint

这个全特征的后门能提供给运营者许多的能力，比如操作文件、注册表和进程，加载额外的组件、执行系统指纹等。

下面是支持的命令列表：

Command number	Description
0	Fingerprint
1	Sets the session ID
2	Creates a process and gets the output (using pipes)
3	Sets the connection retry counter
4	Delays polling time
5	Reads a file or registry key and computes the MD5
6	Creates a process
7	Creates a file, a registry entry or a stream in memory
8	Writes to the registry
9	Queries the registry
10	Searches for files on the system
11	Moves files to another directory
12	Deletes files from the disk
13	List the drives mapped on the system using the <code>GetLogicalDriveStringsW</code> function
14	Creates directory
15	Deletes directory
16	Reads a file from an offset
17	Calls the PE Loader (switch to <code>HTTPprov</code> communication)
18	[Unknown]
19	0: Query a value from the registry, 1: Drop and execute a program
20	Sets an environment variable
21	Runs shellcode in a new thread
22	Retrieves an environment variable
+23 in new version	Restarts itself if the "APPL" environment variable doesn't exist

结论

海莲花组织仍然活跃并在持续更新工具集。这也证明了它挑选目标、限制恶意软件传播和使用不同服务器来避免被关注的意图。Payload解密和侧加载技术仍然不能被检测到，因为这些恶意活动看起来和合法应用没有区别。

Tags: 文件 , 解密 , 加载 , 后门 , 执行 , 合法 , 恶意 , 服务器 , 可执行文件 , 代码 ,

为您推荐了相关的技术文章:

1. [BlackHat 2016 回顾之 JNDI 注入简单解析](#)
2. [unserialize\(\) 实战之 vBulletin 5.x.x 远程代码执行](#)
3. [Rasp 技术介绍与实现](#)
4. [Python Waf黑名单过滤下的一些Bypass思路 - Mosuan's Blog](#)
5. [MS17-010漏洞检测与内网穿透技术的应用 | 浮萍的个人空间](#)

原文链接: www.4hou.com