CREA Practical Exam

Malware Analysis Report

Table of Contents

List of Figures

List of Tables

This Page Was Intentionally Left Blank

1) Executive Summary

On Jan. 12, 2010, the analyst passed the multiple choice exam and proceeded to practical malware analysis of the binary sample provided by IACRB. The purpose of this analysis is to learn what the code does and develop a report which contains the details requested in the “Practical Exam Guide” by IACRB.

The analyst found that the malicious code results in installation of a “Browser Helper Object” (BHO) on the affected system in the form of a “Search Toolbar”. The file targets Mozilla Firefox and Netscape browsers. The code also provides an “Update” capability for the installed toolbar. The search toolbar would allow the user to query search engines such as Google and Yahoo but in the meantime it tracks user search/browsing habits. As a result, it could be classified as a spyware or adware. Internet foot-printing shows that the code is affiliated with the ***MyWebSearch***[1] web site which is known for its adware and spyware activities. The analyst also analyzed new versions of the code not provided by IACRB but downloaded from MyWebSearch. This data reveals similar functionality in the newer versions of the code.

2) Identification

The following table lists the characteristic information about received sample file.

Table 1: Identification results

File name	malware.exe
File Size	49245 bytes
MD5 hash	314e0468433fccd2a52b0b4e192d109b
SHA1 hash	d5d6af77b185612d0c4292ab08929d352fb86b51
SHA256 hash	75b50fbc93248e29efe6ffd506b72bd34f72ecaf2c4138df4acdb0c9b254310e
PEiD	N/A (no known compression/obfuscation found)
PE Structure information	[Base Data] entrypointaddress.: 0x2b72 timedatestamp: 0x463fc0b3 (Tue May 08 00:13:39 2007) machinetype: 0x14c (I386) [sections] name viradd virsiz rawdsiz .text 0x1000 0x5b8e 0x6000 .rdata 0x7000 0x12b7 0x2000 .data 0x9000 0x248 0x1000 .rsrc 0xa000 0x520 0x1000 .reloc 0xb000 0x4f6 0x1000 [directory Information] TLSTable: N/A (No Thread local storage is used) Debug: FOUND, Type=2 (CodeView) PDB File=L:\dev\mws2,2,60,11\Variations\FunWebProductsBar\Release.Plugin\m3Plugin.pdb [6 exports] DllRegisterServer, DllUnregisterServer, NP_GetEntryPoints, NP_Initialize, NP_Shutdown,

[1] http://www.mywebsearch.com

	UTB [7 imports] KERNEL32.dll,USER32.dll,SHELL32.dll, ADVAPI32.dll,ole32.dll,OLEAUT32.dll,VERSION.dll
Version Information	publisher: MyWebSearch.com copyright: Copyright (c) 2000, 2001, 2002, 2003, 2004, 2005, 2006 product: My Web Search Plug-in description: My Web Search Plugin for 32-bit Windows original name: m3Plugin.DLL internal name: MyWebSearch Plugin file version: 1, 0, 1, 3 comments: N/A signers: N/A signing date.: N/A verified: Unsigned

Based on IDA pro FLIRT signatures the file is compiled using MS Visual C++ on Tue May 08 00:13:39 2007. It's a component of a software package named "FunWebProductsBar" from MyWebSearch.com. Courtesy of virustotal.com and/or other sources, the analyst has found the following updated antivirus scan results identifying malicious sample as adware/spyware:

Table 2: Anti-virus detection for the sample

Anti-virus	Detected As
Sunbelt 3.2.1858.2	MyWebSearch Toolbar
NOD32 4796	Win32/Toolbar.MyWebSearch
McAfee 5868	potentially unwanted program MWS
Kaspersky 7.0.0.125	not-a-virus:AdTool.Win32.MyWebSearch

See appendix A for a full copy of anti-virus scan results and signature updates timestamps.

3) Behavioral and code analysis findings

The analyst tested this malicious code in a Windows virtual machine to analyze its behavior and functionality. The operating system displays an "Invalid win32 application" error while running the original file (malware.exe). This indicates that the file can't be run directly and probably needs some kind of "loader". The entry point is set to *DllMain* function which is called when a DLL file is loaded by windows. However, disassembling this function code shows no interesting behavior. The existence of *DllRegisterServer* and *DllUnregisterServer* functions in the export list reveals that the file could be a Windows COM+ DLL file which are normally loaded and registered using regsvr32.exe or rundll32.exe tools. The following commands were used to test this idea on the file that is now renamed to malware.dll:

- Regsvr32 malware.dll
- Rundll32 malware.dll,DllRegisterServer

Invoking the *DllRegisterServer* function using either of the above commands resulted in creation of the following windows registry entry:

Key name: HKEY_LOCAL_MACHINE\SOFTWARE\MyWebSearch\bar

Value(s):

PluginPath=c:\files\

[“c:\files\” is the folder name that contains malware.dll]

No file system or network activities were noticed at this time and it appears that the function did not run completely. The analyst attempted to reverse engineer the detail functionality of DllRegisterServer function in order to discover its detail functionality and find out the reasons of the above incomplete execution.

A combination of static (IDA pro) and runtime analysis (OllyDbg debugger) was used in the reverse engineering process. The following steps were taken to load debug the target DLL inside Ollydbg:

1) Open regsvr32.exe with “c:\files\malware.dll” command line argument
2) Click open and wait until Ollydbg breaks on the entry point
3) Go to Options->Debugging options->Events and set “Break on new module(DLL)”
4) Press F9 until Ollydbg breaks on “c:\files\malware.dll”
5) Go to the malware.dll disassembly, right click and choose “Search for all name(label) in current module” and set a breakpoint on DllRegisterServer
6) Uncheck the “Break on new module(DLL)” and press F9 until it breaks on DllRegisterServer
7) From this point on, you will be able to step into the code line by line

The DllRegisterServer function begins with a call to a custom function sub_1000517 (named *FindBrowserProfileAndUpdateChrome* by the analyst). This function performs the following main tasks:

1) Find the path for special folder “Application Data” by calling *SHGetSpecialFolderLocation* API call

```
.text:10005174 sub_1000517 proc near
……….
.text:10005185          lea    eax, [ebp+appDataPath]
.text:1000518B          push   ebx          ; int
.text:1000518C          push   eax          ; int
.text:1000518D          push   1Ah          ; CSIDL_APPDATA=0x1A={user}\Application Data
.text:1000518F          mov    [ebp+firefoxProfilesINIFile], offset aMozillaFirefox ; "\\Mozilla\\Firefox\\profiles.ini"
.text:10005196          mov    [ebp+netscapeProfilesINIFile], offset aNetscapeNsaePr ; "\\Netscape\\NSAE\\profiles.ini"
.text:1000519D          mov    [ebp+firefoxToolbarChromeName], offset firefoxToolbarChromeName ; "m3ffxtbr"
.text:100051A4          mov    [ebp+netscapeToolbarChromeName], offset netscapeToolbarChromeName ; "m3ntstbr"
.text:100051AB          call   FindAppDataPath
…………
```

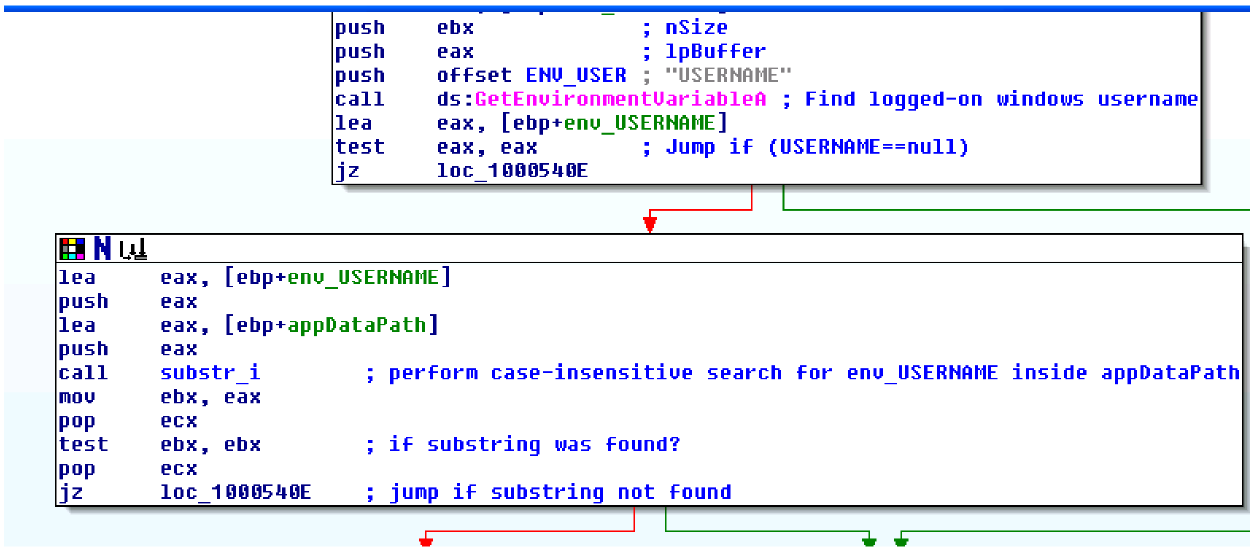
```
.text:10005455 ; int __cdecl FindAppDataPath(int nFolder, int, int)
……….
.text:10005488          lea    eax, [ebp+pidl]
.text:1000548B          push   eax          ; ppidl
.text:1000548C          push   [ebp+nFolder]   ; nFolder=0x1A
```

```
.text:1000548F        push   edi          ; hwndOwner
.text:10005490        call   ds:SHGetSpecialFolderLocation
.text:10005496        test   eax, eax
.text:10005498        jl     short loc_100054DC
……….
```

2) Find current logged-on windows username (via GetEnvironmentVariableA API call), if the "Application Data" folder is matched to the current logged-on user then attempt to update *Mozilla Firefox* or *Netscape Navigator* Chrome registry file[2] (chrome.rdf or installed-chrome.txt) and copy the files listed in the following table to the browser's installation directory (C:\Program Files\Mozilla Firefox\ on analyst's system). This could be the reason for the pervious incomplete execution of the code, since these files were not included in the provided sample by IACRB, and the code assumes that the files exist in its current directory.

Table 3: Files copied by the malware to victim browser directory

Filename	Destination	Description
NPMyWebS.dll	[BrowserInstallationDir]\plugins	Browser plug-in
m3ffxtbr.jar/m3ntstbr.jar	[BrowserInstallationDir]\chrome	Chrome JAR package for Firefox or Netscape
m3ffxtbr.manifest/m3ntstbr.manifest	[BrowserInstallationDir]\chrome	Chrome manifest file for Firefox or Netscape

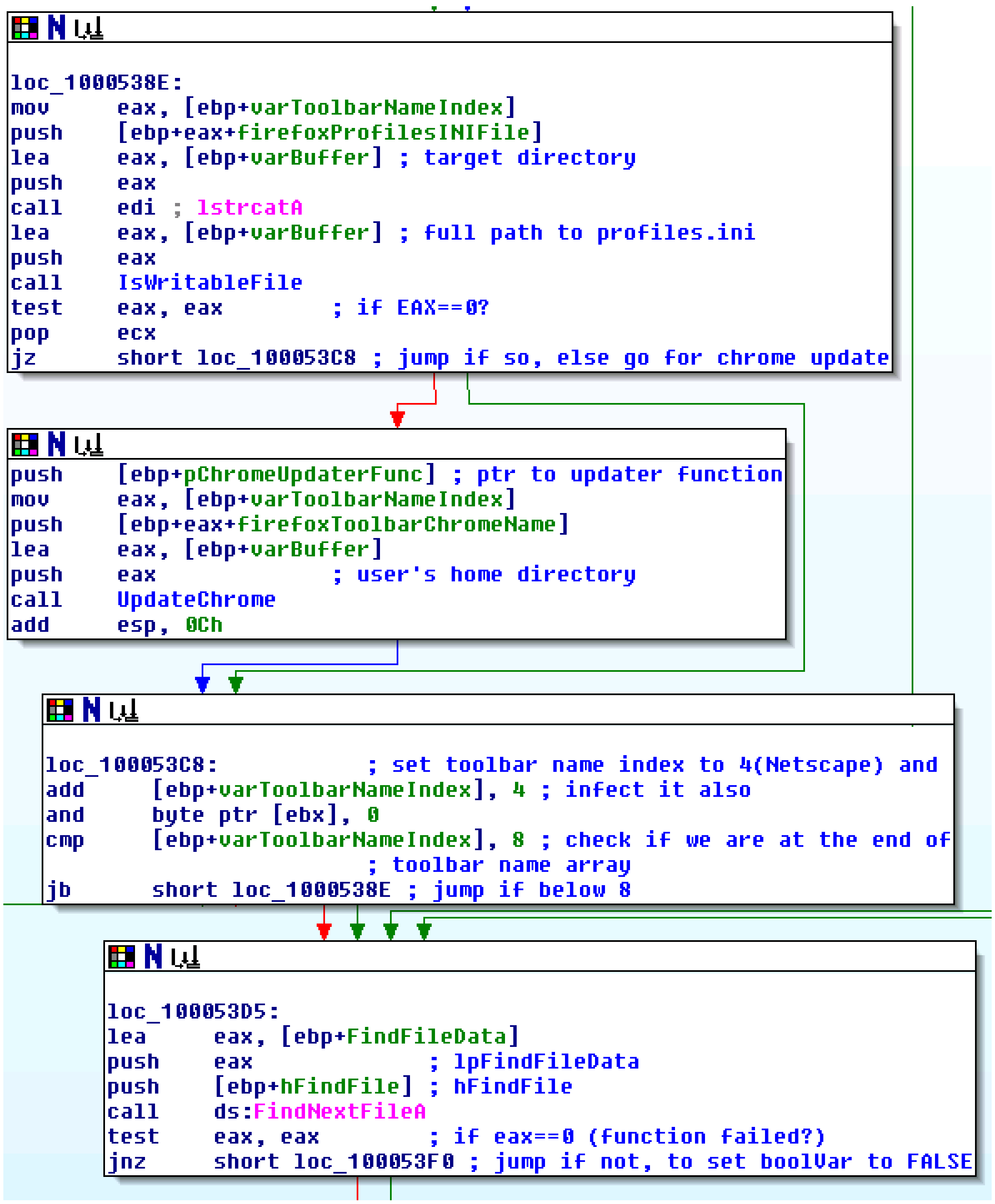

```
push    ebx                ; nSize
push    eax                ; lpBuffer
push    offset ENV_USER ; "USERNAME"
call    ds:GetEnvironmentVariableA ; Find logged-on windows username
lea     eax, [ebp+env_USERNAME]
test    eax, eax           ; Jump if (USERNAME==null)
jz      loc_1000540E
```

```
lea     eax, [ebp+env_USERNAME]
push    eax
lea     eax, [ebp+appDataPath]
push    eax
call    substr_i        ; perform case-insensitive search for env_USERNAME inside appDataPath
mov     ebx, eax
pop     ecx
test    ebx, ebx        ; if substring was found?
pop     ecx
jz      loc_1000540E    ; jump if substring not found
```

Figure 1: validating current windows user and "Application Data" path

If the "Application Data" folder was not valid, the code will attempt to iterate through all subfolders inside "Document and Settings" directory (*FindFirstFileA* and *FindNextFileA* calls), find all writable FireFox and Netscape profiles (custom *IsWritableFile* function), find the chrome registry files and update them to reflect the installation of "MywebSearch" plug-in. This process has been shown in the following reversed code snip:

[2] https://developer.mozilla.org/en/chrome



```
loc_1000538E:
mov     eax, [ebp+varToolbarNameIndex]
push    [ebp+eax+firefoxProfilesINIFile]
lea     eax, [ebp+varBuffer] ; target directory
push    eax
call    edi ; lstrcatA
lea     eax, [ebp+varBuffer] ; full path to profiles.ini
push    eax
call    IsWritableFile
test    eax, eax         ; if EAX==0?
pop     ecx
jz      short loc_100053C8 ; jump if so, else go for chrome update
```

```
push    [ebp+pChromeUpdaterFunc] ; ptr to updater function
mov     eax, [ebp+varToolbarNameIndex]
push    [ebp+eax+firefoxToolbarChromeName]
lea     eax, [ebp+varBuffer]
push    eax              ; user's home directory
call    UpdateChrome
add     esp, 0Ch
```

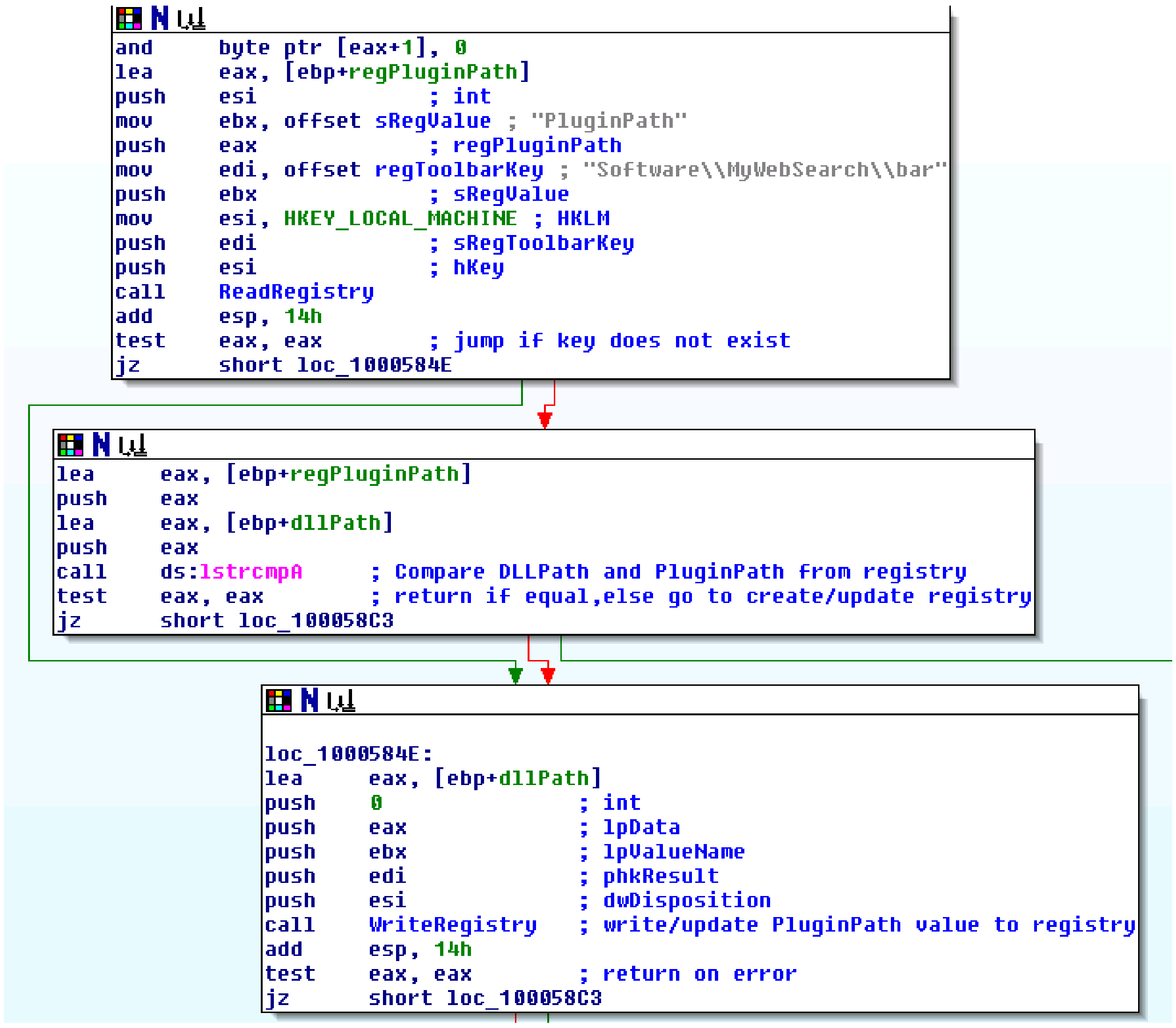
```
loc_100053C8:            ; set toolbar name index to 4(Netscape) and
add     [ebp+varToolbarNameIndex], 4 ; infect it also
and     byte ptr [ebx], 0
cmp     [ebp+varToolbarNameIndex], 8 ; check if we are at the end of
                         ; toolbar name array
jb      short loc_1000538E ; jump if below 8
```

```
loc_100053D5:
lea     eax, [ebp+FindFileData]
push    eax              ; lpFindFileData
push    [ebp+hFindFile] ; hFindFile
call    ds:FindNextFileA
test    eax, eax         ; if eax==0 (function failed?)
jnz     short loc_100053F0 ; jump if not, to set boolVar to FALSE
```

Figure 2: Code to iterate through all user profiles and infect them

The completed reversed code of this function with comments, updated function and local variable names is attached to this report.

After completion of *FindBrowserProfileAndUpdateChrome* function, the code checks windows registry to read “PluginPath” value inside “HKLM\Software\MyWebSearch\bar” key. If the key or value was not found or the read value of “PluginPath” was not equal to current DLL folder name, it attempt to create/update the key and set “PluginPath” value to the current DLL directory name.



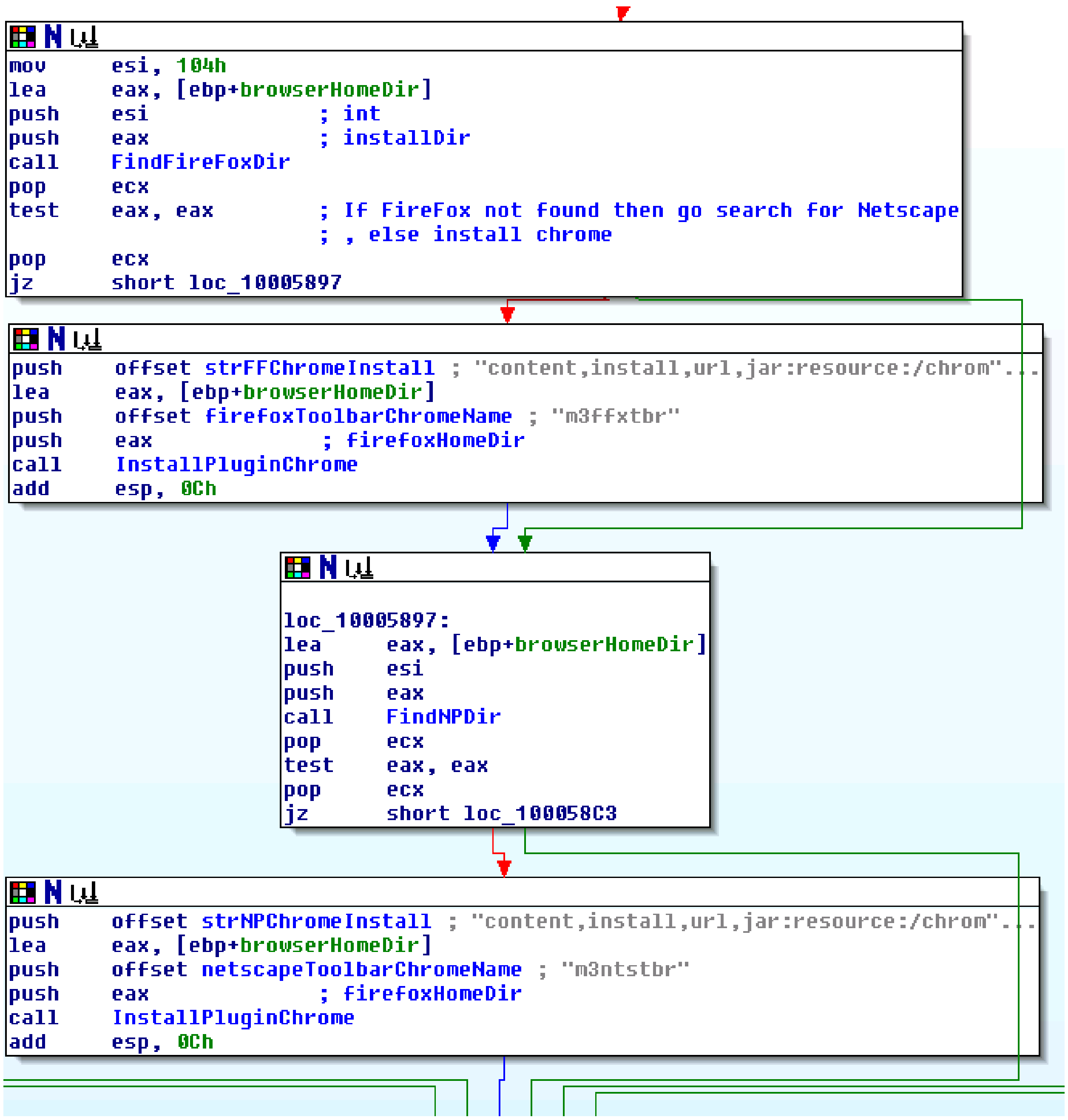
```
and     byte ptr [eax+1], 0
lea     eax, [ebp+regPluginPath]
push    esi             ; int
mov     ebx, offset sRegValue ; "PluginPath"
push    eax             ; regPluginPath
mov     edi, offset regToolbarKey ; "Software\\MyWebSearch\\bar"
push    ebx             ; sRegValue
mov     esi, HKEY_LOCAL_MACHINE ; HKLM
push    edi             ; sRegToolbarKey
push    esi             ; hKey
call    ReadRegistry
add     esp, 14h
test    eax, eax        ; jump if key does not exist
jz      short loc_1000584E
```

```
lea     eax, [ebp+regPluginPath]
push    eax
lea     eax, [ebp+dllPath]
push    eax
call    ds:lstrcmpA     ; Compare DLLPath and PluginPath from registry
test    eax, eax        ; return if equal,else go to create/update registry
jz      short loc_100058C3
```

```
loc_1000584E:
lea     eax, [ebp+dllPath]
push    0                ; int
push    eax              ; lpData
push    ebx              ; lpValueName
push    edi              ; phkResult
push    esi              ; dwDisposition
call    WriteRegistry    ; write/update PluginPath value to registry
add     esp, 14h
test    eax, eax         ; return on error
jz      short loc_100058C3
```

Figure 3: Checking windows registry and updating it if necessary

The execution then continues to final stage of plug-in installation which is copying the plug-in DLL, JAR package and manifest files to victim’s browser’s home directory. The code finds Firefox and Netscape installation directories on the victim’s system and calls a custom function named as *InstallPluginChrome* by the analyst, which is actually responsible to copy the mentioned files and updating installed-chrome.txt file. The following figure shows portion of *DllRegisterServer* function related to this process.



```
mov     esi, 104h
lea     eax, [ebp+browserHomeDir]
push    esi             ; int
push    eax             ; installDir
call    FindFireFoxDir
pop     ecx
test    eax, eax        ; If FireFox not found then go search for Netscape
                        ; , else install chrome
pop     ecx
jz      short loc_10005897
```

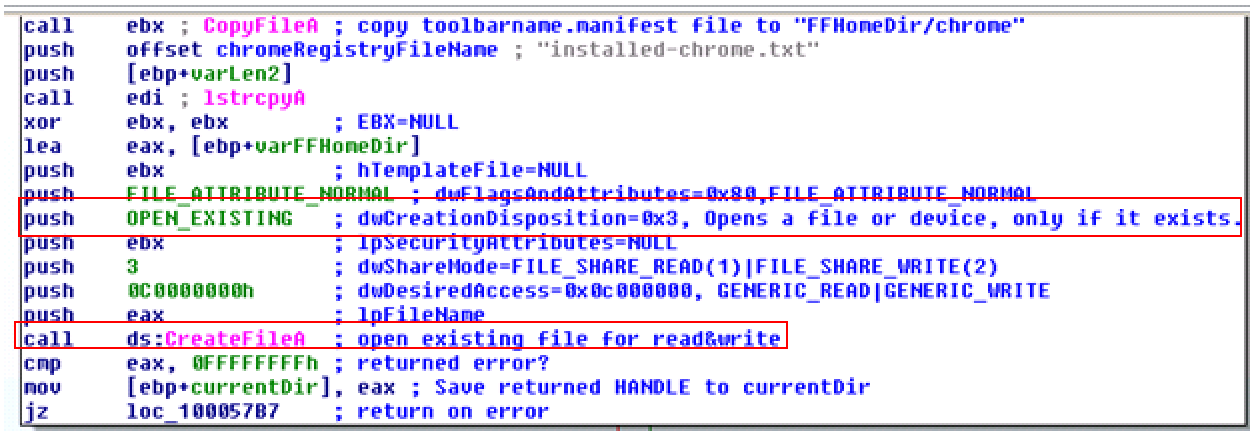
```
push    offset strFFChromeInstall ; "content,install,url,jar:resource:/chrom"...
lea     eax, [ebp+browserHomeDir]
push    offset firefoxToolbarChromeName ; "m3ffxtbr"
push    eax             ; firefoxHomeDir
call    InstallPluginChrome
add     esp, 0Ch
```

```
loc_10005897:
lea     eax, [ebp+browserHomeDir]
push    esi
push    eax
call    FindNPDir
pop     ecx
test    eax, eax
pop     ecx
jz      short loc_100058C3
```

```
push    offset strNPChromeInstall ; "content,install,url,jar:resource:/chrom"...
lea     eax, [ebp+browserHomeDir]
push    offset netscapeToolbarChromeName ; "m3ntstbr"
push    eax             ; firefoxHomeDir
call    InstallPluginChrome
add     esp, 0Ch
```

Figure 4: Browser home directory detection and call to InstallPluginChrome

The analyst also found an issue in the *InstallPluginChrome* function, which can prevent the code from proper plug-in installation on victim systems with Mozilla Firefox version 1.5 or higher. This problem is highlighted in the following code listing of *InstallPluginChrome* function. The code attempts to open "installed-chrome.txt" file using *CreateFileA* API call and sets the *dwCreationDisposition* parameter to OPEN_EXISTING which means that the API call will only succeed if "installed-chrome.txt" file was already present on the victim system. However, this file has been removed from version 1.5 and higher Firefox releases and no longer exists by default (it still can be used to install a new plug-in but it should be created first for this purpose).

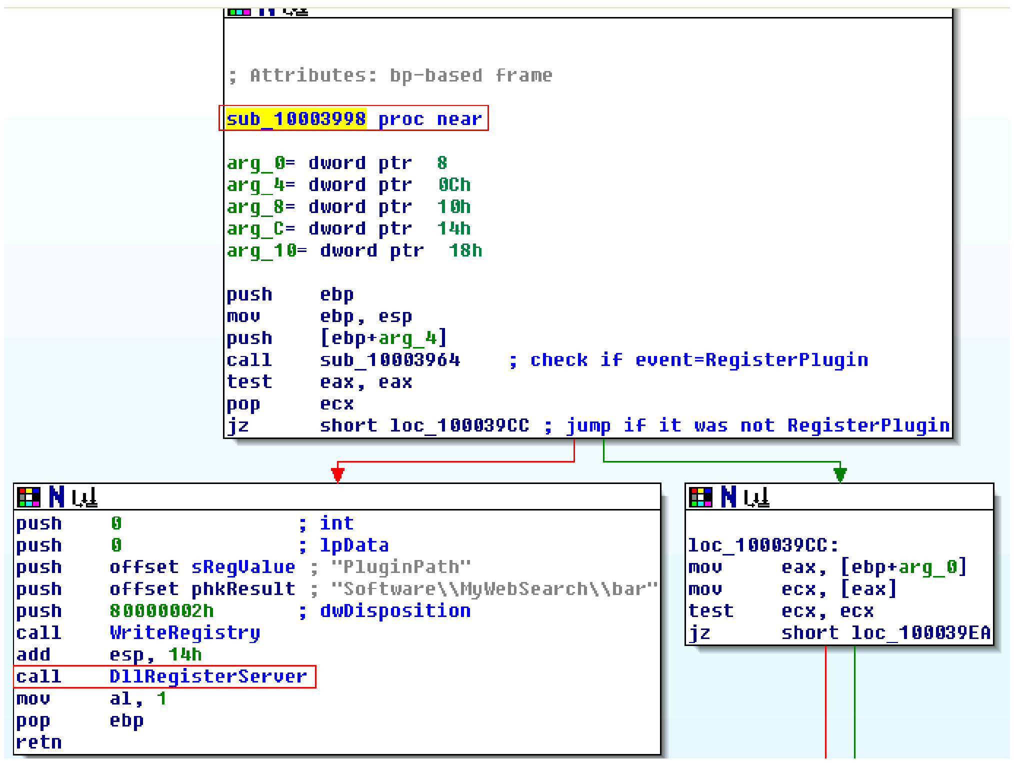


```
call    ebx ; CopyFileA ; copy toolbarname.manifest file to "FFHomeDir/chrome"
push    offset chromeRegistryFileName ; "installed-chrome.txt"
push    [ebp+varLen2]
call    edi ; lstrcpyA
xor     ebx, ebx          ; EBX=NULL
lea     eax, [ebp+varFFHomeDir]
push    ebx               ; hTemplateFile=NULL
push    FILE_ATTRIBUTE_NORMAL ; dwFlagsAndAttributes=0x80,FILE_ATTRIBUTE_NORMAL
push    OPEN_EXISTING     ; dwCreationDisposition=0x3, Opens a file or device, only if it exists.
push    ebx               ; lpSecurityAttributes=NULL
push    3                 ; dwShareMode=FILE_SHARE_READ(1)|FILE_SHARE_WRITE(2)
push    0C0000000h        ; dwDesiredAccess=0x0c000000, GENERIC_READ|GENERIC_WRITE
push    eax               ; lpFileName
call    ds:CreateFileA    ; open existing file for read&write
cmp     eax, 0FFFFFFFFh ; returned error?
mov     [ebp+currentDir], eax ; Save returned HANDLE to currentDir
jz      loc_100057B7      ; return on error
```

Figure 5: Malware programmer wrong assumption about installed-chrome.txt file

A complete reversed and documented code listing of *InstallPluginChrome* is available in the attached IDA pro database file.

Checking the cross references to the *DllRegisterServer* function indicates a single call to it from sub_10003998. Since this function is normally called by a loader tool such as rundll32 rather than other functions in the same COM+ module, it can help the analyst to find the original infection/installation vector of the adware.



```
; Attributes: bp-based frame

sub_10003998 proc near

arg_0= dword ptr  8
arg_4= dword ptr  0Ch
arg_8= dword ptr  10h
arg_C= dword ptr  14h
arg_10= dword ptr  18h

push    ebp
mov     ebp, esp
push    [ebp+arg_4]
call    sub_10003964     ; check if event=RegisterPlugin
test    eax, eax
pop     ecx
jz      short loc_100039CC ; jump if it was not RegisterPlugin
```

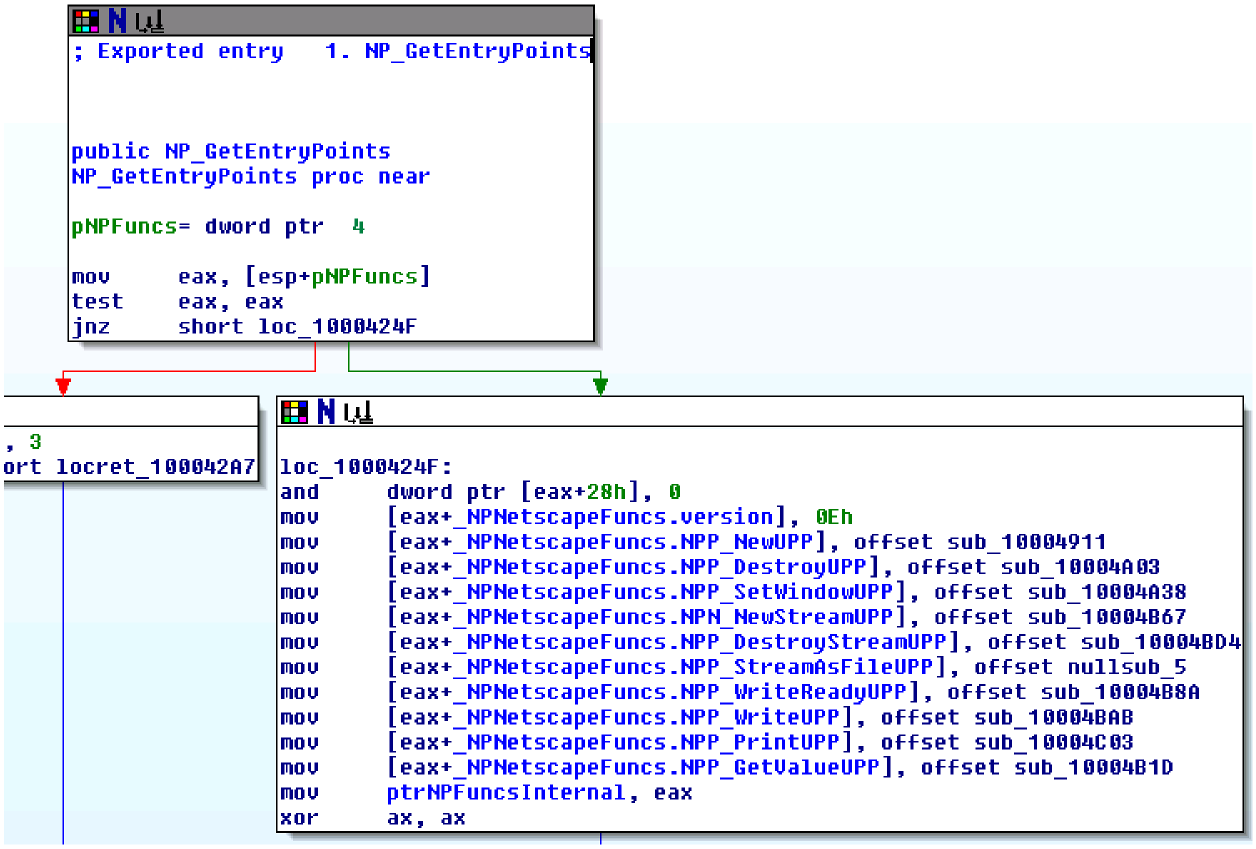
```
push    0                 ; int
push    0                 ; lpData
push    offset sRegValue ; "PluginPath"
push    offset phkResult ; "Software\\MyWebSearch\\bar"
push    80000002h         ; dwDisposition
call    WriteRegistry
add     esp, 14h
call    DllRegisterServer
mov     al, 1
pop     ebp
retn
```

```
loc_100039CC:
mov     eax, [ebp+arg_0]
mov     ecx, [eax]
test    ecx, ecx
jz      short loc_100039EA
```

Figure 6: Call to DllRegisterServer from plug-in event handler

Code analysis shows that sub_10003998 will be called indirectly by the browser after the plug-in has been loaded. The exported *NP_GetEntryPoints* function is responsible to set function pointers that results in sub_10003998 being called. The following figure shows code listing of this function. The

_NPNetscapeFuncs structure (Added to IDA pro structures by the analyst) is passed by browser to this function and the function sets functions pointers inside this structure to the proper functions in its code.



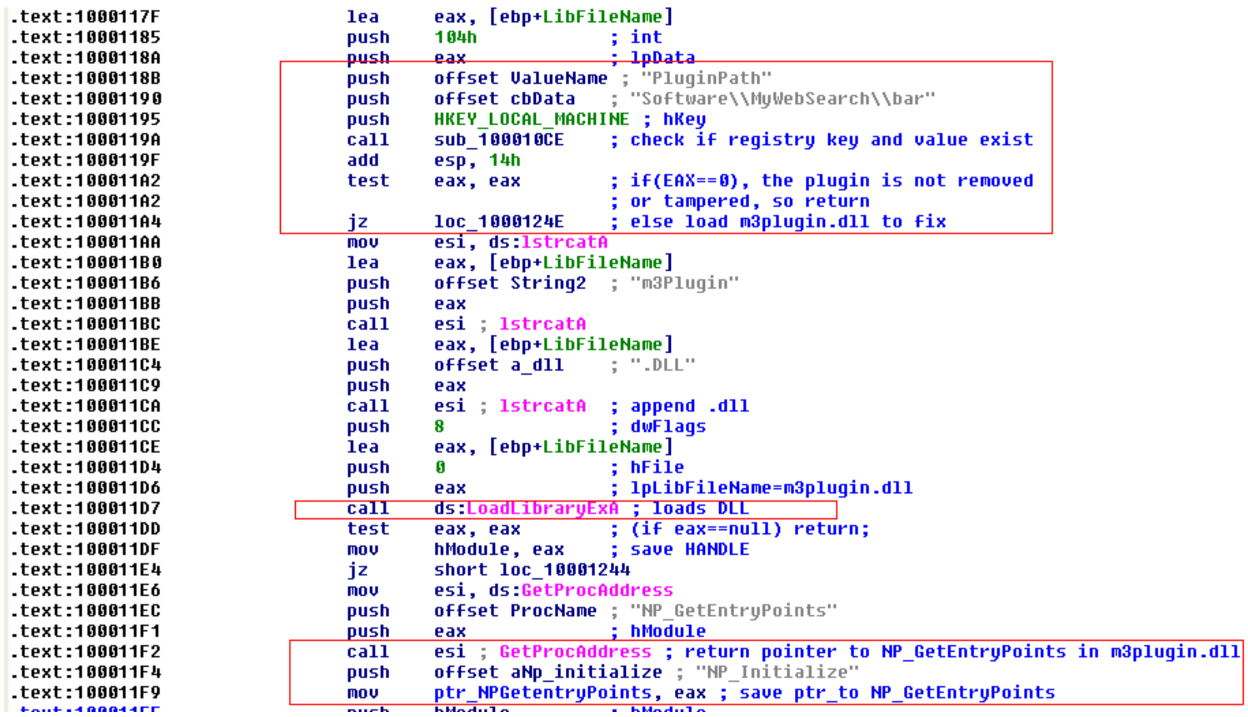
```
; Exported entry   1. NP_GetEntryPoints

public NP_GetEntryPoints
NP_GetEntryPoints proc near

pNPFuncs= dword ptr  4

mov     eax, [esp+pNPFuncs]
test    eax, eax
jnz     short loc_1000424F
```

```
, 3
ort locret_100042A7
```

```
loc_1000424F:
and     dword ptr [eax+28h], 0
mov     [eax+_NPNetscapeFuncs.version], 0Eh
mov     [eax+_NPNetscapeFuncs.NPP_NewUPP], offset sub_10004911
mov     [eax+_NPNetscapeFuncs.NPP_DestroyUPP], offset sub_10004A03
mov     [eax+_NPNetscapeFuncs.NPP_SetWindowUPP], offset sub_10004A38
mov     [eax+_NPNetscapeFuncs.NPN_NewStreamUPP], offset sub_10004B67
mov     [eax+_NPNetscapeFuncs.NPP_DestroyStreamUPP], offset sub_10004BD4
mov     [eax+_NPNetscapeFuncs.NPP_StreamAsFileUPP], offset nullsub_5
mov     [eax+_NPNetscapeFuncs.NPP_WriteReadyUPP], offset sub_10004B8A
mov     [eax+_NPNetscapeFuncs.NPP_WriteUPP], offset sub_10004BAB
mov     [eax+_NPNetscapeFuncs.NPP_PrintUPP], offset sub_10004C03
mov     [eax+_NPNetscapeFuncs.NPP_GetValueUPP], offset sub_10004B1D
mov     ptrNPFuncsInternal, eax
xor     ax, ax
```

Figure 7: Plug-in function pointer setups by NP_GetEntryPoints

Based on Mozilla developer's documentation, *NP_GetEntryPoints* function will be called by the browser immediately after *NP_Initilize*, but since it was found that the current code copies another DLL file (NPMyWebS.dll) to browser's plug-ins directory, the browser actually calls *NP_GetEntryPoints* of NPMyWebS.dll and not the above one. The analyst downloaded a recent version of MyWebSearch toolbar (Firefox version) from the company's web site, installed it on a VM and reversed the *NP_GetEntryPoints* function inside NPMyWebS.dll file. It was found that the actual plug-in DLL (NPMyWebS.dll) first checks the windows registry for MyWebSearch key and value. If they are not found then it dynamically loads m3plugin.dll (the current code under analysis) and calls its *NP_GetEntryPoints function*. This process has been shown in the following code listing.



```
.text:1000117F                 lea     eax, [ebp+LibFileName]
.text:10001185                 push    104h            ; int
.text:1000118A                 push    eax             ; lpData
.text:1000118B                 push    offset ValueName ; "PluginPath"
.text:10001190                 push    offset cbData   ; "Software\\MyWebSearch\\bar"
.text:10001195                 push    HKEY_LOCAL_MACHINE ; hKey
.text:1000119A                 call    sub_100010CE    ; check if registry key and value exist
.text:1000119F                 add     esp, 14h
.text:100011A2                 test    eax, eax        ; if(EAX==0), the plugin is not removed
.text:100011A2                                         ; or tampered, so return
.text:100011A4                 jz      loc_1000124E    ; else load m3plugin.dll to fix
.text:100011AA                 mov     esi, ds:lstrcatA
.text:100011B0                 lea     eax, [ebp+LibFileName]
.text:100011B6                 push    offset String2  ; "m3Plugin"
.text:100011BB                 push    eax
.text:100011BC                 call    esi ; lstrcatA
.text:100011BE                 lea     eax, [ebp+LibFileName]
.text:100011C4                 push    offset a_dll    ; ".DLL"
.text:100011C9                 push    eax
.text:100011CA                 call    esi ; lstrcatA  ; append .dll
.text:100011CC                 push    8               ; dwFlags
.text:100011CE                 lea     eax, [ebp+LibFileName]
.text:100011D4                 push    0               ; hFile
.text:100011D6                 push    eax             ; lpLibFileName=m3plugin.dll
.text:100011D7                 call    ds:LoadLibraryExA ; loads DLL
.text:100011DD                 test    eax, eax        ; (if eax==null) return;
.text:100011DF                 mov     hModule, eax    ; save HANDLE
.text:100011E4                 jz      short loc_10001244
.text:100011E6                 mov     esi, ds:GetProcAddress
.text:100011EC                 push    offset ProcName ; "NP_GetEntryPoints"
.text:100011F1                 push    eax             ; hModule
.text:100011F2                 call    esi ; GetProcAddress ; return pointer to NP_GetEntryPoints in m3plugin.dll
.text:100011F4                 push    offset aNp_initialize ; "NP_Initialize"
.text:100011F9                 mov     ptr_NPGetentryPoints, eax ; save ptr_to NP_GetEntryPoints
```

Figure 8: NPMyWebS.dll dynamically loads malware and calls its NP_GetEntryPoints function

The complete code listing of the above process is included in Appendix B.

Based on the above finding, it's now clear that the code under analysis (malware.dll or m3plugin.dll) is a form of watchdog code that is loaded by the main search toolbar plug-in(NPMyWebS.dll) to detect any tamper or removal of the toolbar and re-install it again.

Another interesting exported function in malware.dll is UTB. The function simply pushes the address of a custom function named *restoreToolbarFunc* and calls *FindBrowserProfileAndUpdateChrome* which was analyzed before. Analysis of the *restoreToolbarFunc* indicates that it reads the "*localstore.rdf*"[3] file inside victim user's application data directory. This file stores customized data on plug-ins such as visibility, size and sort orders. The function finds the entry for *MyWebSearch* toolbar inside this file and checks if the "collapsed" property is set to "true" which means the toolbar won't be displayed. If so, this property is set back to "false". This is another measure by the adware to prevent users from simply unloading the toolbar from "View->toolbar" menu option in Mozilla Firefox. The adware probably would add the following windows registry entry to HKLM\Software\Microsoft\Windows\CurrentVersion\Run which will restore the toolbar on every system startup:

Rundll32.exe malware.dll, UTB

The attached IDA pro file has areversed code listing of *restoreToolbarFunc.*

The following table summarizes all the above behavioral and code analysis findings with regards to CREA practical instructions document:

[3] http://kb.mozillazine.org/Localstore.rdf

Table 4: Analysis findings

General function of the malware	The malware sample was found to be a "watchdog" program for "MyWebSearch" Adware. It provides the following main functionalities: 1. Once loaded by the browser toolbar plug-in (NPMyWebS.dll), it will re-copy the adware search toolbar files from the current directory and update the related windows registry key. This behavior is intended to counter anti-spyware or manual removal/tamper 2. Through another exported function, the malware code can check if the victim user has chosen not to view the search toolbar in the browser. In this case, the code attempts to overwrite this setting and make the toolbar visible again.
File System activity	**Files created**: c:\program files\Mozilla Firefox\plugins\NPMyWebS.dll c:\program files\Mozilla Firefox\chrome\m3ffxtbr.jar (m3ntstbr.jar) c:\program files\Mozilla Firefox\chrome\ m3ffxtbr.manifest (m3ntstbr.manifest) **Files Accessed/Modified**: c:\program files\Mozilla Firefox\chrome\installed-chrome.txt c:\program files\Mozilla Firefox\chrome\overlayinfo\browser\content\overlays.rdf c:\Documents and Settings\[username]\Application Data\Mozilla\Firefox\Profiles\[profile_name]\chrome\chrome.rdf C:\Documents and Settings\[username]\Application Data\Mozilla\Firefox\Profiles\[profile_name]\localstore.rdf
Windows Registry activity	**Modified or created entries**: Key name: HKLM\Software\MyWebSearch\bar Value: PluginPath=[folder path of the malware]
Network activity	No network activity was noticed
Original infection vector	The code is a component of "MyWebSearch" browser toolbar that is normally downloaded and installed by the victim who thinks it's a harmless program. After installation, the program would add the following windows registry key to check and fix the toolbar visibility status : Key name:HKLM\Software\Microsoft\Windows\CurrentVersion\Run Value: somename=rundll32.exe malware.dll, UTB
Information about development of the malware	The Adware has been developed by MyWebSearch.com. Another adware software named "FunWebProducts" could be installed subsequently on the victim system that has already search toolbar installed. The malware code is compiled using MS Visual C++ on Tue May 08 00:13:39 2007

4) Mitigation

The analyst recommends following steps to remove the "MyWebSearch" toolbar from an infected system:

1) Close the Firefox or Netscape browser and remove NPMyWebS.dll file from [browser home dir]\plugins directory
2) Remove m3ffxtbr.jar(m3ntstbr.jar for Netscape) and m3ffxtbr.manifest(m3ntstbr.manifest for Netscape) from chrome [browser home dir]\chrome folder
3) Edit and remove the following lines from installed-chrome.txt:
   content,install,url,jar:resource:/chrome/m3ffxtbr.jar!/ or
   content,install,url,jar:resource:/chrome/m3ntstbr.jar!/
4) Edit and remove any entries for "MyWebSearch" found inside localstore.rdf

5) Remove the following line from overlays.rdf
   chrome://m3ffxtbr/content/menu.xul

Appendix A- Full Anti-virus Scan results

Antivirus	Version	Last Update	Result
a-squared	4.5.0.50	2010.01.22	Riskware.AdTool.Win32.MyWebSearch!IK
AhnLab-V3	5.0.0.2	2010.01.22	-
AntiVir	7.9.1.146	2010.01.22	-
Antiy-AVL	2.0.3.7	2010.01.22	AdTool/Win32.MyWebSearch.gen
Authentium	5.2.0.5	2010.01.22	W32/HackTool.UP
Avast	4.8.1351.0	2010.01.22	-
AVG	9.0.0.730	2010.01.22	-
BitDefender	7.2	2010.01.22	-
CAT-QuickHeal	10.00	2010.01.22	-
ClamAV	0.94.1	2010.01.22	Adware.Search-53
Comodo	3669	2010.01.22	ApplicUnwnt.Win32.Toolbar.MyWebSearch
DrWeb	5.0.1.12222	2010.01.22	Adware.Msearch
eSafe	7.0.17.0	2010.01.21	AdTool.Win32.MyWebSe
eTrust-Vet	None	2010.01.22	-
F-Prot	4.5.1.85	2010.01.21	W32/HackTool.UP
F-Secure	9.0.15370.0	2010.01.22	-
Fortinet	4.0.14.0	2010.01.22	Misc/Mywebsearch
GData	19	2010.01.22	-
Ikarus	T3.1.1.80.0	2010.01.22	not-a-virus:AdTool.Win32.MyWebSearch
Jiangmin	13.0.900	2010.01.22	-
K7AntiVirus	7.10.951	2010.01.20	not-a-virus:AdTool.Win32.MyWebSearch
Kaspersky	7.0.0.125	2010.01.22	not-a-virus:WebToolbar.Win32.MyWebSearch.as
McAfee	5868	2010.01.21	potentially unwanted program MWS

McAfee+Artemis	5868	2010.01.21	potentially unwanted program MWS
McAfee-GW-Edition	6.8.5	2010.01.22	-
Microsoft	1.5405	2010.01.22	-
NOD32	4796	2010.01.22	Win32/Toolbar.MyWebSearch
Norman	6.04.03	2010.01.21	-
nProtect	2009.1.8.0	2010.01.22	Trojan-Clicker/W32.MyWebSearch.49245
Panda	10.0.2.2	2010.01.22	-
PCTools	7.0.3.5	2010.01.22	-
Prevx	3.0	2010.01.22	Medium Risk Malware
Rising	22.31.04.04	2010.01.22	Adware.MyWebSearch.f
Sophos	4.50.0	2010.01.22	-
Sunbelt	3.2.1858.2	2010.01.22	MyWebSearch Toolbar
Symantec	20091.2.0.41	2010.01.22	-
TheHacker	6.5.0.9.158	2010.01.22	-
TrendMicro	9.120.0.1004	2010.01.22	-
VBA32	3.12.12.1	2010.01.21	-
ViRobot	2010.1.22.2151	2010.01.22	Not_a_virus:AdTool.MyWebSearch.49245
VirusBuster	5.0.21.0	2010.01.21	-

Appendix B – NP_GetEntryPoints code from recent version of NPMywebS.dll

```
.text:10001175
.text:10001175              public NP_GetEntryPoints
.text:10001175 NP_GetEntryPoints proc near
.text:10001175
.text:10001175 LibFileName     = byte ptr -104h
.text:10001175 pNPFuncs        = dword ptr  8
.text:10001175
.text:10001175              push    ebp
.text:10001176              mov     ebp, esp
.text:10001178              sub     esp, 104h
.text:1000117E              push    esi
.text:1000117F              lea     eax, [ebp+LibFileName]
.text:10001185              push    104h            ; buffer size
.text:1000118A              push    eax             ; lpData
.text:1000118B              push    offset ValueName ; "PluginPath"
.text:10001190              push    offset cbData   ; "Software\\MyWebSearch\\bar"
.text:10001195              push    HKEY_LOCAL_MACHINE ; hKey
.text:1000119A              call    sub_100010CE    ; check if mywebsearch registry key and value exist
.text:1000119F              add     esp, 14h
.text:100011A2              test    eax, eax        ; if(EAX==0), the plug-in is not removed  or tampered, so return
.text:100011A4              jz      loc_1000124E    ; else load m3plugin.dll to fix
.text:100011AA              mov     esi, ds:lstrcatA
.text:100011B0              lea     eax, [ebp+LibFileName]
.text:100011B6              push    offset String2  ; "m3Plugin"
.text:100011BB              push    eax
.text:100011BC              call    esi ; lstrcatA
.text:100011BE              lea     eax, [ebp+LibFileName]
.text:100011C4              push    offset a_dll    ; ".DLL"
.text:100011C9              push    eax
.text:100011CA              call    esi ; lstrcatA  ; appends .dll
.text:100011CC              push    8               ; dwFlags
.text:100011CE              lea     eax, [ebp+LibFileName]
.text:100011D4              push    0               ; hFile
.text:100011D6              push    eax             ; lpLibFileName=m3plugin.dll
.text:100011D7              call    ds:LoadLibraryExA ; loads DLL
.text:100011DD              test    eax, eax        ; (if eax==null) return;
.text:100011DF              mov     hModule, eax    ; save HANDLE
.text:100011E4              jz      short loc_10001244
.text:100011E6              mov     esi, ds:GetProcAddress
.text:100011EC              push    offset ProcName ; "NP_GetEntryPoints"
.text:100011F1              push    eax             ; hModule
.text:100011F2              call    esi ; GetProcAddress ; return pointer to NP_GetEntryPoints in m3plugin.dll
.text:100011F4              push    offset aNp_initialize ; "NP_Initialize"
.text:100011F9              mov     ptr_NPGetentryPoints, eax ; save ptr_to NP_GetEntryPoints
.text:100011FE              push    hModule         ; hModule
.text:10001204              call    esi ; GetProcAddress
.text:10001206              push    offset aNp_shutdown ; "NP_Shutdown"
.text:1000120B              mov     ptr_NPInitialize, eax ; save ptr_to NP_Initialize
```

```
.text:10001210            push    hModule         ; hModule
.text:10001216            call    esi ; GetProcAddress
.text:10001218            cmp     hModule, 0
.text:1000121F            mov     ptr_NPShutdown, eax
.text:10001224            jz      short loc_10001244
.text:10001226            mov     ecx, ptr_NPGetentryPoints
.text:1000122C            test    ecx, ecx      ;check if ptr was not null
.text:1000122E            jz      short loc_10001244
.text:10001230            cmp     ptr_NPInitialize, 0
.text:10001237            jz      short loc_10001244
.text:10001239            test    eax, eax
.text:1000123B            jz      short loc_10001244
.text:1000123D            push    [ebp+pNPFuncs]
.text:10001240            call    ecx ; calls NPGetentryPoints from m3plugin.dll
.text:10001242            jmp     short loc_10001251
.text:10001244 ; ---------------------------------------------------------------------------
.text:10001244
.text:10001244 loc_10001244:
.text:10001244            mov     ecx, offset hModule
.text:10001249            call    sub_10001256     ;unload m3plugin.dll
.text:1000124E
.text:1000124E loc_1000124E:
.text:1000124E            push    4
.text:10001250            pop     eax
.text:10001251
.text:10001251 loc_10001251:
.text:10001251            pop     esi
.text:10001252            leave
.text:10001253            retn    4
.text:10001253 NP_GetEntryPoints endp
```