

Analysis Report (TLP:WHITE)

Analysis of a PlugX variant (PlugX version 7.0)

Conducted by CIRCL - Computer Incident Response Center Luxembourg
Team CIRCL

March 29, 2013

Document version: 1.0



CIRCL

Computer Incident
Response Center
Luxembourg

Contents

1	Scope of work	3
2	Analyzed samples	3
2.1	Limitations	6
2.2	Sharing	6
3	Executive summary	6
4	Analysis	6
4.1	Techniques used	6
4.2	Execution process	7
4.2.1	Diagram	7
4.2.2	Explanation	7
4.3	Implemented commands	10
4.4	Command details	12
4.4.1	Option	12
4.4.2	Disk	12
4.4.3	Screen	12
4.4.4	Process	12
4.4.5	Service	12
4.4.6	Shell	12
4.4.7	Telnet	13
4.4.8	RegEdit	13
4.4.9	Nethood	13
4.4.10	Portmap	13
4.4.11	SQL	13
4.4.12	Netstat	13
4.4.13	Keylogger	13
4.5	Other notable commands and functions	13
4.5.1	log	13
4.6	Persistency	15
4.7	Control	15
4.8	Network and domain information	16
4.8.1	Network	16
4.8.2	Domain	17
4.9	Current version and history of PlugX	18
A	Appendix	18
A.1	Indicators of Compromise (IOC)	18
A.1.1	Pipes	18
A.1.2	Files and directories	18
A.1.3	Registry	19
A.1.4	Network (hostname and destination IP addresses)	19
A.2	References	19
A.3	VirusTotal results	20

1 Scope of work

This report is the analysis of a Remote Access Tool (RAT) which we call a variant of Plugx¹. Plugx is an interesting piece of malware for several reasons:

- It demonstrates the attack principle of the fastest/cheapest path of attack² by abusing perfectly valid signed binaries to perform the attack
- It features ways to defeat other protection mechanisms like UAC³
- In contrast to many other pieces of malware, the author⁴ shows the ability to write good code, especially doing logging the right way to improve the piece of software
- It appears to be modularized and easily extensible

2 Analyzed samples

- Sample A - Stage 1 of Malware
 - Description
 - * Hash found in a malware database
 - Original filename
 - * update.exe
 - Hashes
 - * MD5: f1f48360f95e1b43e9fba0fec5a2afb8
 - * SHA1: 70ceb467db7b0161d22e4545479f747417b9705a
 - * SHA-256: 2bc5ce39dd9afe2157448d3f6d8cb9c549ed39543d159616e38480b9e6c11c49
 - Filetype
 - * PE32 executable (GUI) Intel 80386, for MS Windows, RAR self-extracting archive
 - Filesize
 - * 370702 Bytes (326KB)
 - Compile time
 - * Sat Jun 9 15:19:49 2012
- Sample B - Valid, signed McAfee binary
 - Description
 - * File dropped by Sample A
 - Original filename
 - * mcvsmapi.exe
 - Hashes

¹Known variant names: Gulpix, Korplug

²<http://satoss.uni.lu/seminars/srm/pdfs/2012-Alexandre-Dulaunoy.pdf>

³<http://msdn.microsoft.com/en-us/library/windows/desktop/bb756996.aspx>

⁴For better readability we do not distinguish between a single author or a group of authors. Hence the expression is a synonym for "the authors"

- * MD5: 4e1e0b8b0673937415599bf2f24c44ad
- * SHA1: 9224de3af2a246011c6294f64f27206d165317ba
- * SHA-256: ae16e10e621d6610a3f7f2c7122f9d1263700ba02d1b90e42798decbb2fe84096
- Filetype
 - * PE32 executable (GUI) Intel 80386, for MS Windows
- Filesize
 - * 262672 Bytes (257K)
- Compile time
 - * Fri May 8 17:59:52 2009
- Authenticode⁵ verification

```
1 Verified: Signed
2 Signers:
3 McAfee, Inc.
4 VeriSign Class 3 Code Signing 2004 CA
5 Class 3 Public Primary Certification Authority
6 Signing date: 5:24 PM 5/8/2009
7 Publisher: McAfee, Inc.
8 Description: McAfee VirusMap Reporting module
9 Product: McAfee VirusScan API
10 Version: 13,11,0,0
11 File version: 13,11,102,0
12 Strong Name: Unsigned
13 Original Name: McVsMap.EXE
14 Internal Name: McVsMap
15 Copyright: Copyright 2008 McAfee, Inc.
16 Comments: n/a
17 MD5: 4e1e0b8b0673937415599bf2f24c44ad
18 SHA1: 9224de3af2a246011c6294f64f27206d165317ba
19 SHA256: ae16e10e621d6610a3f7f2c7122f9d1263700ba02d1b90e42798decbb2fe84096
```

- Sample C - DLL to be loaded by Sample B, loads Sample D
 - Description
 - * File dropped by Sample A
 - Original filename
 - * McUtil.DLL
 - Hashes
 - * MD5: ad4a646b38a482cc07d5b09b4fffd3b3
 - * SHA1: ae0f9bf2740d00c5d485827eb32aca33feaa3a90
 - * SHA-256: 0a99238e1ebebc47d7a89b2ccddfae537479f7f77322b5d4941315d3f7e5ca48
 - Filetype
 - * PE32 executable (DLL) (GUI) Intel 80386, for MS Windows
 - Filesize
 - * 49152 Bytes (48K)

⁵<http://msdn.microsoft.com/en-us/library/ms537359%28v=vs.85%29.aspx>

- Compile time
 - * Wed Mar 13 02:52:28 2013
- Sample D - Malicious payload to be loaded by Sample C
 - Description
 - * File dropped by Sample A
 - Original filename
 - * McUtil.DLL.PPT
 - Hashes
 - * MD5: 545bb4365a9b7cdb6d22844ebeeadda93
 - * SHA1: a267f1183b4ff843d68a63264846abf78cc71d1f
 - * SHA-256: d4fe890a08d4dd44b58a3b85b2a7e89536338099c1c42a9b7e85f4007b0a37b7
 - Filetype
 - * pure code (IA32) without headers
 - Filesize
 - * 124820 Bytes (122K)
 - Compile time
 - * unknown (pure code)
- Sample E - Stage 2 of Malware
 - Description
 - * Extracted malware from memory
 - Original filename
 - * dump00C60000.bin
 - Hashes
 - * MD5: 65ceb039e7b4731a165cfee081e220af
 - * SHA1: b49766187971e3070644a9de2054bc93241b2263
 - * SHA-256: deecac56026f3804968348c8afa5b7aba10900aeabee05751c0fcac2b88cff71e
 - Filetype
 - * PE32 executable (DLL) (GUI) Intel 80386, for MS Windows
 - Filesize
 - * 176128 Bytes (172K)
 - Compile time
 - * Mon Nov 26 04:46:01 2012
- Sample F - UAC circumvention
 - Description
 - * File temporarily created on filesystem
 - Original filename

- * UAC.TMP
- Hashes
 - * MD5: 52df5c2c07433e2a8f054c2347acb3b4
 - * SHA1: 8051474c1fc0d8f404a42ea32eca1699e54f02e1
 - * SHA-256: dc09091e5d0ce03c6144748f17bd636f2f0b2ca56f88b550c1d48860596dbdb1
- Filetype
 - * PE32 executable (DLL) (GUI) Intel 80386, for MS Windows
- Filesize
 - * 2560 Bytes (2.5K)
- Compile time
 - * Thu Mar 29 08:03:43 2012

2.1 Limitations

This work has been done with utmost care, following best practices in software reversing, forensic investigations and/or information gathering. However, the work is only covering small aspects (based on the indicators given, lacking full context) and not an exhaustive analysis, and hence the report is as-is, not giving any guarantees of completeness or claiming absolute accuracy. This work is provided for information only.

2.2 Sharing

The document is classified as TLP:WHITE, CIRCL authorizes everyone to share this analysis report as-is without modification.

3 Executive summary

The analyzed malicious software is an exhaustive Remote Access Tool (RAT) that defeats several protection methods of modern Windows operating systems, including execution of signed code and defeating UAC in Windows 7. It comes with a multitude of functionalities that are well implemented.

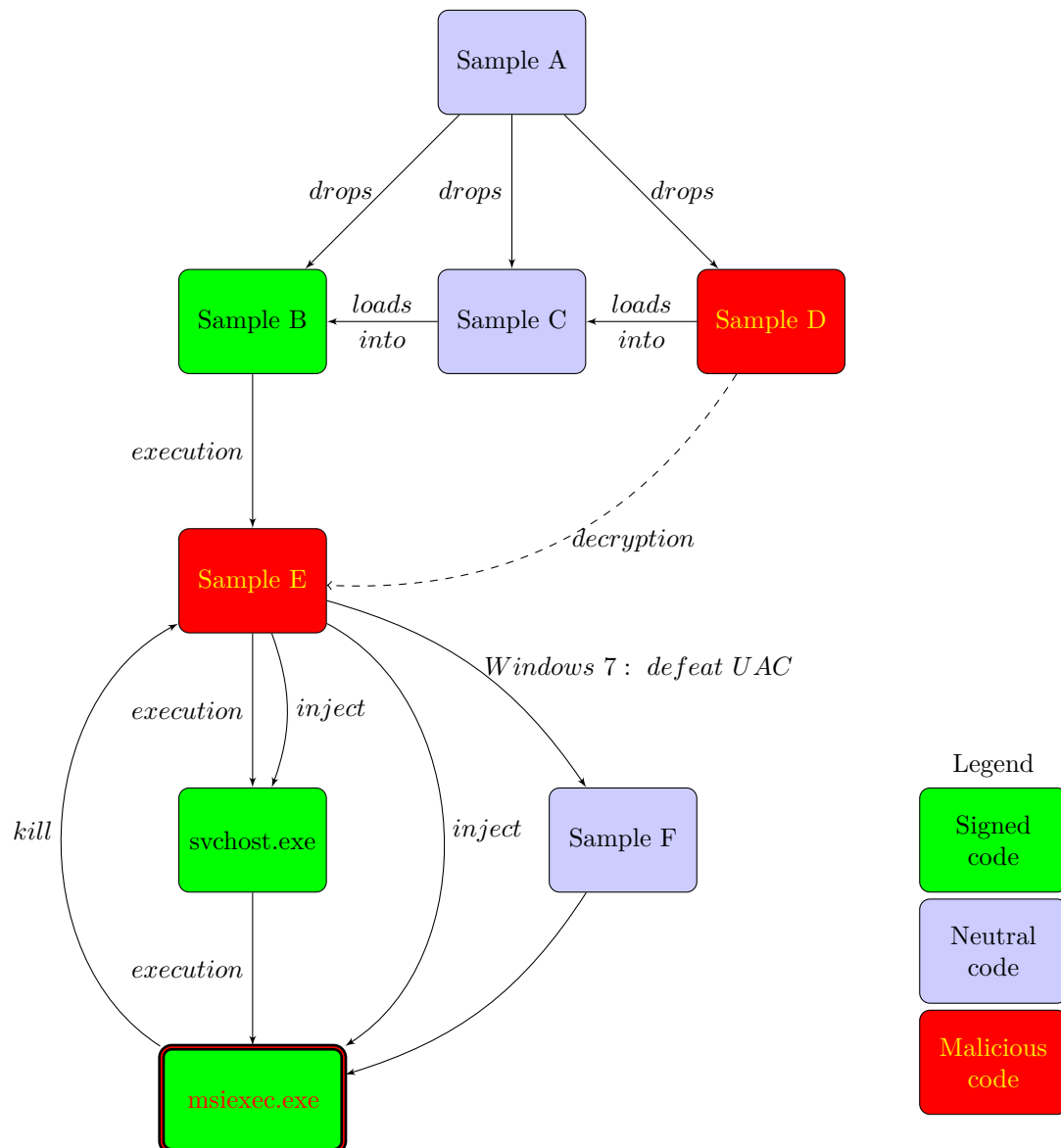
4 Analysis

4.1 Techniques used

The analysis has been done using a mixed-approach of dynamic analysis and static analysis in order to overcome some of the obfuscation and encryptions used by the malware. Some of the techniques might have also an impact on the interpretation of the malware. Unfortunately, when we started this investigation, the IP address is no longer accepting connections on the given ports when tested on 2013-03-26. An interaction following the protocol of this malware is therefore no longer possible.

4.2 Execution process

4.2.1 Diagram



4.2.2 Explanation

Sample A is a self-extracting archive which contains three files, Sample B, Sample C and Sample D. It is assumed that Sample A is a part of another attack vector, like PDF or Office document attacks where the user just opens a crafted document which exploits the document reader, drops and opens both a readable document and a malicious file like Sample A.

¹ Type = Rar

² Solid = –

3	Blocks = 3					
4	Multivolume = -					
5	Volumes = 1					
6						
7	Date	Time	Attr	Size	Compressed	Name
8						
9	2009-05-14	00:56:12	A	262672	119784	mcvsmap.exe
10	2013-03-13	09:52:28	A	49152	20285	McUtil.DLL
11	2013-03-13	14:56:12	A	124820	124820	McUtil.DLL.PPT
12						
13				436644	264889	3 files , 0 folders

Executing the self-extracting archive extracts the files and runs mcvsmap.exe (Sample B). Sample B is a valid signed file that the author of the malware took from a software bundle from McAfee. Sample B, when executed, attempts to load a file McUtil.DLL from the same directory, which usually is another component of McAfee. The malware author instead bundled the valid McAfee file Sample B with a custom DLL (Sample C). Since the file will be loaded without hesitation (there are no protection mechanisms implemented; neither does McAfee check if the imported file meets any conditions nor is any protection implemented for loading unsigned libraries in signed code), the signed Sample B jumps into the beginning of the code section of Sample C (via Push/Return):

```

.text:0040465E add     esp, 0C0h
.text:00404661 lea     eax, [ebp+88h+LibFileName] ; McUtil.dll
.text:00404664 push    eax ; lpLibFileName
.text:00404665 call    ds:LoadLibraryA
.text:00404668 push    0FFFFFFFh
.text:00404670 push    offset start_of_mciutil_dll
.text:00404675 retn
.text:00404676 ; -----

```

At the target location the following code is executed:

```

1 read_execute_file()
2 {
3     NumberOfBytesRead = GetModuleFileNameW(hModule, &filename, 0x2000u);
4     lstrcatW(&filename, L".PPT");
5     hFile_mcutil.dll.ppt = CreateFileW(&filename, GENERIC_READ, 1u, 0, OPEN_EXISTING, 0,
6         0);
7     if ( hFile_mcutil.dll.ppt == -1 )
8     {
9         result = GetLastError();
10    }
11    else
12    {
13        buffer = VirtualAlloc(0, 0x100000u, MEM_COMMIT, PAGE_EXECUTE_READWRITE);
14        if ( buffer && ReadFile(hFile_mcutil.dll.ppt, buffer, 0x100000u, &NumberOfBytesRead, 0) )
15        {
16            CloseHandle(hFile_mcutil.dll.ppt);
17            buffer();
18            Sleep(0xFFFFFFFF);
19            Sleep(0xFFFFFFFF);
20            Sleep(0xFFFFFFFF);
21            result = 0;
22        }
23        else
24        {
25            result = GetLastError();
26        }
27    }
28 }

```



```

25     }
26   }
27   return result;
28 }

```

The code retrieves the filename of itself (line 3), which is McUtil.DLL, and appends .PPT (line 4). A handle to the filename McUtil.DLL.PPT is created in line 5. In line 12 an executable memory region is created, which is filled with the content of the file McUtil.DLL.PPT (line 13). After closing the handle to the file (line 15), the memory region is called (line 16). The next screenshot shows that the memory contains only pure code without any overhead like MZ/PE headers. The entropy of this file is 7.997904 bits per byte:

```

00C40000 cmp     edx, 0CBE1F4B7h
00C40006 test    edx, 0FF2A2004h
00C4000C xor     esi, 0BBE5876Fh
00C40012 dec     edx
00C40013 xor     edx, 2EA8D3E3h
00C40019 cmp     edi, 45B286E4h
00C4001F and     esi, 0A05A96A4h
00C40025 inc     edx
00C40026 mov     edi, 2BFCC370h
00C4002B cmp     edi, 0A347050Bh
00C40031 cmp     esi, 3646EDD4h
00C40037 mov     edx, 9C72B29Ah
00C4003C call    $+5
00C40041 test    esi, 760A483Dh
00C40047 dec     esi
00C40048 and     edx, 0BB03BC9h
00C4004E or      edi, 0BE8D653Ah
00C40054 and     edi, 0CF84EF63h
00C4005A or      esi, 44A352FBh
00C40060 and     edx, 231692BEh
00C40066 and     edi, 92FA77C5h
00C4006C and     edi, 0C537FF3Bh
00C40072 pop     eax
00C40073 add     esi, 5F409FE1h
00C40079 dec     edi
00C4007A or      edx, 0F77945A9h
00C40080 cmp     edx, 64CCDD6Dh
00C40086 add     esi, 808BF208h
00C4008C cmp     edx, 6CD05232h
00C40092 dec     esi
00C40093 cmp     edx, 0F3F3B05Ah
00C40099 mov     edi, 49319D2Ah
00C4009E sub     edx, 0F80B7589h
00C400A4 cmp     esi, 0F5C87C7Ah
00C400AA add     edx, 0FC22F58Eh

```

The code, when executed, reveals the first hint about what we found:

```

00C4069E mov     eax, [ebp+arg_0]
00C406A1 mov     [esi+10h], eax
00C406A4 mov     eax, [ebp+arg_10]
00C406A7 mov     [esi+14h], eax
00C406AA mov     eax, [ebp+arg_14]
00C406AD mov     [esi+18h], eax
00C406B0 mov     dword ptr [esi], 'PLUG'
00C406B6 mov     eax, [edi+28h]
00C406B9 add     eax, esi
00C406BB mov     [esi+1Ch], eax

```

It decompresses and decrypts itself, using the Microsoft API call RtlDecompressBuffer and the custom decryption routine:

```

1  int crypt(unsigned int a1, int a2, int a3, int a4)
2  {
3      if ( a4 > 0 )
4      {
5          v10 = a3 - a2;
6          do
7          {
8              a1 = a1 + (a1 >> 3) - 0x11111111;
9              a1 = a1 + (a1 >> 5) - 0x22222222;
10             a1 += 0x44444444 - (a1 << 9);
11             a1 += 0x33333333 - (a1 << 7);
12             v7 = *(v10 + a2++) ^ (a1 + a1 + a1 + a1);
13             v8 = a4 - 1;
14             *(a2 - 1) = v7;
15         }
16         while ( !v8 );
17     }
18     return 0;
19 }

```

The decrypted and decompressed file is not written onto disk, it always remains in memory. Sample E is the extracted version of this memory segment. At this point it can be mentioned that neither the encrypted Sample D nor the decrypted memory segment Sample E are detected by Virus scanners.

After some initialisation work like adjusting tokens (SeDebugPrivilege, SeTcbPrivilege⁶, to act as part of the operating system), a new process is started, the original svchost.exe from Microsoft, and the code from Sample E is injected into the memory of that process. In a next step, svchost.exe is instructed to execute the original msixec.exe from Microsoft, where also memory is injected like it has been done for svchost.exe. Special conditions apply when run under Window 7, which is protected by User Account Control (UAC). UAC is supposed to protect the user better from running malware by requesting the administrator for approval before running a potentially dangerous application. In the environment of Windows 7, the malware drops temporarily file Sample F, which it uses to evade or defeat the UAC mechanism. After killing the parent processes, only two processes are left: svchost and msixec. Both are verified binaries, none of the includes a malicious DLL.

svchost.exe	2760	0.37	4,596 K (Verified) Microsoft Windows C:\Windows\System32\svchost.exe Host Process for Windows S...
msixec.exe	2340	0.17	2,112 K (Verified) Microsoft Windows C:\Windows\System32\msixec.exe Windows® installer

Nevertheless, they both contain the malicious code. At this point in time the malware is already talking to the C&C, no user interaction was required, all standard security mechanisms were defeated.

4.3 Implemented commands

The analysis of Sample B revealed the commands as shown in the table below:

⁶<http://technet.microsoft.com/en-us/library/bb457125.aspx>

Table 1: Implemented commands

Source file	Internal command	subcommand	Description
XPlugOption.cpp	Option	0x2000 0x2001 0x2002 0x2003 0x2005	lock workstation shutdown workstation (forced) reboot workstation shutdown workstation (graceful) show messagebox
XPlugDisk.cpp	Disk	0x3000 0x3001 0x3002 0x300A 0x3004 0x3007 0x300D 0x300C 0x300E	enumerate drives find file find file recursively create directory read file write file file copy/rename/delete/move create process on hidden desktop get expanded environment string
XPlugScreen.cpp	Screen	0x4000 0x4004 0x4005 0x4006 0x4100	Remote Desktop capabilities send mouse event send keyboard event send CTRL-Alt-Delete take screenshot
XPlugProcess.cpp	Process	0x5000 0x5001 0x5002	create process enumerate processes kill process
XPlugService.cpp	Service	0x6000 0x6001 0x6002 0x6003 0x6004	query service config change service config (forced) start service control service delete service
XPlugShell.cpp	Shell	0x7002	start a cmd shell
XPlugTelnet.cpp	Telnet	0x7100	start telnet server
XPlugRegedit.cpp	RegEdit	0x9000 0x9001 0x9002 0x9003 0x9004 0x9005 0x9006 0x9007	enumerate keys create key delete key copy key enumerate values set value delete value get value
XPlugNethood.cpp	Nethood	0xA000	enumerate network resources
XPlugPortMap.cpp	Portmap	0xB000	starts port mapping
XPlugSQL.cpp	SQL	0xC000 0xC001 0xC002	get data source information get driver description execute statement
XPlugNetstat.cpp	Netstat	0xD000 0xD001 0xD002	get TCP table get UDP table set TCP entry
XPlugKeyLogger.cpp	Keylogger	0xE000	starts key logger thread

4.4 Command details

4.4.1 Option

XPlugOption implements commands to lock the workstation, shut it down or reboot it. In addition, XPlugOption can create a thread that calls `MessageBoxW()` in order to present a message box to the user.

4.4.2 Disk

XPlugDisk is used to enumerate connected disk drives and can be used to find and manipulate files and directories. In addition, XPlugDisk can be used to create a process, optionally on a hidden Windows desktop with the name "HH", as the code below illustrates:

```
1  if ( a1->hidden )
2  {
3      hDesktop = CreateDesktopW(L"HH", 0, 0, 0, 0x10000000u, 0);
4      if ( !hDesktop )
5          log("XPlugDisk.cpp", 665, 0);
6  }
7  hidden = a1->hidden;
8  StartupInfo.lpDesktop = (hidden != 0 ? L"HH" : 0);
9  StartupInfo.cb = 68;
10 StartupInfo.dwFlags = 1;
11 StartupInfo.wShowWindow = hidden == 0;
12 if ( CreateProcessW(0, &a1->commandline, 0, 0, 0, 0, 0, 0, &StartupInfo, &
    ProcessInformation) )
13 {
14     ...
15 }
```

4.4.3 Screen

XPlugScreen is not only taking screenshots, it is also implementing remote desktop capabilities. It is able to capture the screen (internal command: `ScreenT1`) and can send mouse and keyboard events (internal command: `ScreenT2`).

4.4.4 Process

XPlugProcess implements three commands and is able to enumerate, create and kill processes.

4.4.5 Service

In the module XPlugService commands are available related to Windows services. Code is implemented to query service configurations, change service configuration, start, control and delete services.

4.4.6 Shell

A remote shell for the attacker is created in the module XPlugShell, by creating an asynchronous set of pipes (`\pipe\`a and `\pipe\`b) for `cmd.exe` and the console attached to it (`AttachConsole()`).

4.4.7 Telnet

cmd.exe /Q is executed in the module XPlugTelnet in order to start a telnet server on the attacked machine.

4.4.8 RegEdit

XPlugRegedit implements a set of commands to process the Windows registry. It is able to enumerate, create, delete and copy keys. It is also able to enumerate, set, delete and get values from the registry.

4.4.9 Nethood

XPlugNethood is the module to enumerate network resources like network shares.

4.4.10 Portmap

XPlugPortMap indicates that it performs some port mapping, however, the code is not understood, yet.

4.4.11 SQL

XPlugSQL implements three functions to query SQL servers: a function to get data source information, a function to get the driver description and a function to execute SQL statements.

4.4.12 Netstat

XPlugNetstat gets the TCP and UDP connection table and is able to set TCP table entries.

4.4.13 Keylogger

The keylogger implemented in XPlugKeyLogger catches Window titles, date, time and logs entered keys into the file

```
1 C:\Documents and Settings\All Users\VirusMap\NvSmart.hlp
```

It has the format following the example below:

```
1 2013-03-26 09:40:57 | C:\Program Files\Mozilla Firefox\firefox.exe - Mozilla Firefox
2 www.google.com
3
4 2013-03-26 09:47:49 | C:\WINDOWS\system32\notepad.exe | Untitled - Notepad
5 This is not a password
6
7 2013-03-26 09:48:06 | C:\WINDOWS\Explorer.EXE | C:\Documents and Settings\All Users\
  VirusMap
```

4.5 Other notable commands and functions

4.5.1 log

This function is called almost everywhere when the author expects that a functions returns an error, at 1036 places. This is obviously done to ensure code quality.

```

1 write_log(LPCWSIR lpBuffer)
2 {
3     ExpandEnvironmentStringsW(L"%ALLUSERSPROFILE%", &path_to_bug.log, 0x800u);
4     // %ALLUSERSPROFILE%\SxS\bug.log
5     lstrcatW(&path_to_bug.log, L"\\SxS");
6     CreateDirectoryW(&path_to_bug.log, 0);
7     SetFileAttributesW(&path_to_bug.log, 6u);
8     lstrcatW(&path_to_bug.log, L"\\bug.log");
9     result = CreateFileW(&path_to_bug.log, 0x40000000u, 1u, 0, 4u, 2u, 0);
10    if ( result != -1 )
11    {
12        if ( SetFilePointer(result, 0, 0, 2u) != -1 )
13        {
14            GetLocalTime(&SystemTime);
15            NumberOfBytesWritten = wsprintfW(
16                &Buffer,
17                L"%4.4d-%2.2d-%2.2d %2.2d:%2.2d:%2.2d: ",
18                SystemTime.wYear,
19                SystemTime.wMonth,
20                SystemTime.wDay,
21                SystemTime.wHour,
22                SystemTime.wMinute,
23                SystemTime.wSecond);
24            if ( WriteFile(result, &Buffer, 2 * NumberOfBytesWritten, &NumberOfBytesWritten,
25                0) )
26            {
27                len = lstrlenW(lpBuffer);
28                WriteFile(result, lpBuffer, 2 * len, &len, 0);
29            }
30            result = CloseHandle(result);
31        }
32        return result;
33    }

```

Example log file entries from file

```

1 %ALLUSERSPROFILE%\SxS\bug.log

1 2013-03-25 11:43:28: file: XSetting.h, line: 57, error: [1300]Not all privileges
   referenced are assigned to the caller.
2 2013-03-25 11:51:12: file: XInstallUAC.cpp, line: 162, error: [5]Access is denied.
3 2013-03-25 13:59:45: file: XRIL.cpp, line: 186, error: [1300]Not all privileges
   referenced are assigned to the caller.
4 2013-03-25 14:07:12: file: XRIL.cpp, line: 186, error: [123]The filename, directory
   name, or volume label syntax is incorrect.
5 2013-03-25 14:07:12: file: XSetting.h, line: 58, error: [3]The system cannot find the
   path specified.
6 2013-03-25 14:21:12: file: dllmain.cpp, line: 47, error: [1300]Not all privileges
   referenced are assigned to the caller.
7 2013-03-25 17:31:58: file: XInstall.cpp, line: 451, error: [5]Access is denied.
8 2013-03-25 17:37:00: file: XSoTcpHttp.cpp, line: 646, error: [12029]*

```

In addition an exception filter is installed to fetch the circumstances of otherwise not caught errors:

```

1 TopLevelExceptionHandler(struct _a1_30 *a1)
2 {
3     ...
4     if ( wsprintfA(
5         &OutputString,

```

```

6      "EName:%s,EAddr:0x%p,ECode:0x%p,EAX:%p,EBX:%p,ECX:%p,EDX:%p,ESI:%p,EDI:%p,EBP
7      :%p,ESP:%p,EIP:%p\r\n",
8      &String1,
9      a1->ECode[3],
10     *a1->ECode,
11     v6->reg_eax,
12     v6->reg_ebx,
13     v6->reg_ecx,
14     v6->reg_edx,
15     v6->reg_esi,
16     v6->reg_edi,
17     v6->reg_ebp,
18     v6->reg_esp,
19     v6->reg_eip) >= 256 )
20     log("XException.cpp", 39, 0);
21     call_write_log(&OutputString);
22     call_OutputDebugStringA(&OutputString);
23     ...
24 }

```

4.6 Persistency

The three files Sample B, C and D are copied into the directory

```
1 C:\Documents and Settings\All Users\VirusMap
```

respectively in

```
1 C:\ProgramData\VirusMap (C:\Users\All Users\VirusMap)
```

After that, a new registry entry is set:

```
1 HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run
```

which calls mcvsmap.exe (Sample B) after login.

Another option is the installation as a service in

```
1 HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\VirusMap
```

The key "ImagePath" calls the same binary mcvsmap.exe (Sample B).

4.7 Control

The attacked computer uses TCP and UDP to connect to port 443 on help.yahoo-upgrade.com (122.199.194.197). Unfortunately, the machine at that IP address doesn't seem to reply to our requests anymore on 2013-03-26.

The Passive DNS showed some other associated domains and hostnames with this IP address:

```

1 help.yahoo-upgrade.com
2 support.yahoo-upgrade.com
3 update.ayuisyahooapis.com
4 support.ayuisyahooapis.com
5 update.trendmicrosoft.co.in

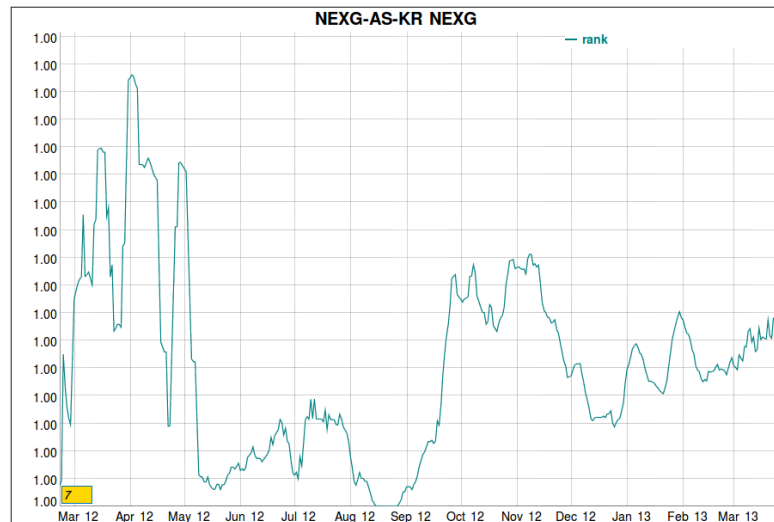
```

It's highly probable that these hostnames were also used for other campaigns. You might use these as additional indicators for the detection of potential infections.

4.8 Network and domain information

4.8.1 Network

The IP address is located in the ASN 17877 and the ISP is not a known bulletproof hoster as you can see on its historical⁷ BGP ranking evolution.



```

1 inetnum:      122.199.128.0 - 122.199.255.255
2 netname:      VAAN
3 descr:        NexG
4 descr:        5F SeoulAcademy B/D, 967-6 Daechi-Dong, Gangnam-Gu, 135-280
5 descr:        *****
6 descr:        Allocated to KRNIC Member.
7 descr:        If you would like to find assignment
8 descr:        information in detail please refer to
9 descr:        the KRNIC Whois Database at:
10 descr:       "http://whois.nic.or.kr/english/index.html"
11 descr:       *****
12 country:     KR
13 admin-c:     SL1625-AP
14 tech-c:      SL1625-AP
15 remarks:     www.nexg.net
16 status:      ALLOCATED PORTABLE
17 mnt-by:      MNT-KRNIC-AP
18 mnt-lower:   MNT-KRNIC-AP
19 changed:     hm-changed@apnic.net 20060606
20 source:      APNIC
21
22 person:      Sanguk Lee
23 nic-hdl:     SL1625-AP
24 e-mail:      ip@nexg.net
25 address:     5F SeoulAcademy B/D, 967-6 Daechi-Dong, Gangnam-Gu, 135-280
26 phone:       +82-2-538-7060
27 fax-no:      +82-2-571-8998
28 country:     KR
29 changed:     hostmaster@nida.or.kr 20050105

```

⁷http://bgpranking.circl.lu/asn_details?asn=17877


```
30 mnt-by: MNT-KRNIC-AP
31 source: APNIC
32
33 inetnum: 122.199.128.0 - 122.199.255.255
34 netname: VAAN-KR
35 descr: NexG
36 country: KR
37 admin-c: LS151-KR
38 tech-c: LS151-KR
39 status: ALLOCATED PORTABLE
40 mnt-by: MNT-KRNIC-AP
41 mnt-irt: IRT-KRNIC-KR
42 remarks: This information has been partially mirrored by APNIC from
43 remarks: KRNIC. To obtain more specific information, please use the
44 remarks: KRNIC whois server at whois.krnic.net.
45 changed: hostmaster@nic.or.kr
46 source: KRNIC
```

4.8.2 Domain

```
1 Domain Name: YAHOO-UPGRADE.COM
2 Registrar: JIANGSU BANGNING SCIENCE & TECHNOLOGY CO. LTD
3 Whois Server: whois.55hl.com
4 Referral URL: http://www.55hl.com
5 Name Server: DNS5.4CUN.COM
6 Name Server: DNS6.4CUN.COM
7 Status: ok
8 Updated Date: 08-aug-2012
9 Creation Date: 18-jul-2011
10 Expiration Date: 18-jul-2013
11
12 >>> Last update of whois database: Wed, 27 Mar 2013 22:36:13 UTC <<<
13
14 Domain Name: yahoo-upgrade.com
15
16 Registrant Contact:
17 yahoo
18 yahoo yahoo whiteyoo_123@yahoo.com
19 telephone: +48.56756756756
20 fax: +48.56732453453
21 yahoo yahoo yahoo 345345
22 CA
23
24 Administrative Contact:
25 yahoo yahoo whiteyoo_123@yahoo.com
26 telephone: +48.56756756756
27 fax: +48.56732453453
28 yahoo yahoo yahoo 345345
29 CA
30
31 Technical Contact:
32 yahoo yahoo whiteyoo_123@yahoo.com
33 telephone: +48.56756756756
34 fax: +48.56732453453
35 yahoo yahoo yahoo 345345
36 CA
37
38 Billing Contact:
39 yahoo yahoo whiteyoo_123@yahoo.com
40 telephone: +48.56756756756
```

```
41 fax: +48.56732453453
42 yahoo yahoo yahoo 345345
43 CA
```

4.9 Current version and history of PlugX

A version string can be found in this binary:

```
1 d:\work\plug7.0(mcvsmapi)(fking)j()\shellcode\shellcode\XPlug.h
```

This could mean PlugX, version 7.0 codename fking, build for mcvsmap. References can be found on the internet for previous versions of this malware family:

```
1 d:\work\plug4.0(nvsmapi)(sxl)\shellcode\shellcode\XPlug.h
2 d:\work\plug3.1(icesword)\shellcode\shellcode\XPlug.h
3 d:\work\Plug3.0(Gf)UDP\Shell6\Release\Shell6.pdb
4 i:\work\plug2.0()\shellcode\shellcode\XPlug.h
```

A Appendix

A.1 Indicators of Compromise (IOC)

This section summarizes the known indicators of compromise. The list might not be exhaustive, but the existence of any or all of the following indicators might help to discover an infection.

A.1.1 Pipes

```
1 \PIPE\a$PID
2 \PIPE\b$PID
3 \PIPE\RUN_AS_USER($PID)
```

(where \$PID is the process ID of the active malicious process)

A.1.2 Files and directories

- Static files (dropped files)

– update.exe

```
1 MD5:      f1f48360f95e1b43e9fba0fec5a2afb8
2 SHA1:      70ceb467db7b0161d22e4545479f747417b9705a
3 SHA-256:   2bc5ce39dd9afe2157448d3f6d8cb9c549ed39543d159616e38480b9e6c11c49
```

– mcvsmap.exe

```
1 MD5:      4e1e0b8b0673937415599bf2f24c44ad
2 SHA1:      9224de3af2a246011c6294f64f27206d165317ba
3 SHA-256:   ae16e10e621d6610a3f7f2c7122f9d1263700ba02d1b90e42798decbb2fe84096
```

– McUtil.DLL

```
1 MD5:      ad4a646b38a482cc07d5b09b4fffd3b3
2 SHA1:      ae0f9bf2740d00c5d485827eb32aca33feaa3a90
3 SHA-256:   0a99238e1ebecb47d7a89b2ccddfae537479f7f77322b5d4941315d3f7e5ca48
```

– McUtil.DLL.PPT

1	MD5:	545bb4365a9b7cdb6d22844ebeedda93
2	SHA1:	a267f1183b4ff843d68a63264846abf78cc71d1f
3	SHA-256:	d4fe890a08d4dd44b58a3b85b2a7e89536338099c1c42a9b7e85f4007b0a37b7

– UAC.TMP

1	MD5:	52df5c2c07433e2a8f054c2347acb3b4
2	SHA1:	8051474c1fc0d8f404a42ea32eca1699e54f02e1
3	SHA-256:	dc09091e5d0ce03c6144748f17bd636f2f0b2ca56f88b550c1d48860596dbdb1

Files and/or directories might be hidden and carry the system flag

1	C:\ProgramData\VirusMap (Windows 7)
2	C:\Users\All Users\VirusMap (Windows 7)
3	C:\Documents and Settings\All Users\VirusMap (Windows XP)
4	%ALLUSERSPROFILE%\SxS\bug.log
5	C:\Documents and Settings\All Users\VirusMap\NvSmart.hlp

A.1.3 Registry

1	HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\VirusMap and a key referencing Sample B
2	HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run and a key referencing Sample B

A.1.4 Network (hostname and destination IP addresses)

1	help.yahoo--upgrade.com
2	122.199.194.197

A.2 References

- WHITE PAPER: PLUG X - PAYLOAD EXTRACTION
 - http://www.contextis.com/files/PlugX_-_Payload_Extraction_March_2013_1.pdf
 - Context Information Security - <http://www.contextis.com/>
 - Published 2013-03-22
- An Analysis of PlugX
 - <http://lastline.com/blog.php>
 - Lastline - <http://www.lastline.com/>
 - no publication date found
- PlugX is becoming mature
 - http://www.securelist.com/en/blog/208193974/PlugX_is_becoming_mature
 - Kaspersky Lab - <http://www.kaspersky.com/>

- Published 2012-11-27
- Unplugging PlugX Capabilities
 - <http://blog.trendmicro.com/trendlabs-security-intelligence/unplugging-plugx-capabilities>
 - TrendMicro - <http://www.trendmicro.eu/>
 - Published 2012-09-17
- Tracking down the author of the PlugX RAT
 - <http://labs.alienvault.com/labs/index.php/2012/tracking-down-the-author-of-the-plugx-rat>
 - AlienVault - <http://labs.alienvault.com>
 - Published 2012-09-13

A.3 VirusTotal results

- Sample A

1	MicroWorld-eScan:	Trojan.Agent.AZDK
2	nProtect:	Trojan.Agent.AZDK
3	McAfee:	RDN/Generic BackDoor!gq
4	Malwarebytes:	Trojan.Dropper.CH
5	Symantec:	WS.Reputation.1
6	Norman:	Agent.APIJH
7	TrendMicro-HouseCall:	BKDR_POISON.PQ
8	Avast:	Win32:Gulpix-B [Trj]
9	Kaspersky:	Backdoor.Win32.Gulpix.bo
10	BitDefender:	Trojan.Agent.AZDK
11	Agnitum:	Backdoor.Gulpix!EFaRR6zLtc4
12	ViRobot:	Backdoor.Win32.A.Gulpix.370702.B
13	Comodo:	UnclassifiedMalware
14	F-Secure:	Trojan.Agent.AZDK
15	DrWeb:	Trojan.Click2.52215
16	VIPRE:	Trojan.Win32.Generic!BT
17	AntiVir:	TR/Agent.azdk.3
18	TrendMicro:	BKDR_POISON.PQ
19	McAfee-GW-Edition:	RDN/Generic BackDoor!gq
20	Sophos:	Troj/Agent-AATT
21	Kingsoft:	Win32.Hack.Gulpix.(kcloud)
22	Microsoft:	Backdoor:Win32/Plugx.A
23	GData:	Trojan.Agent.AZDK
24	AhnLab-V3:	Backdoor/Win32.Gulpix
25	Ikarus:	Backdoor.Win32.Gulpix
26	Fortinet:	W32/Gulpix.BO!tr.bdr
27	AVG:	Agent4.AKAP
28	Panda:	Trj/CI.A
29	Scanned: 2013-03-21 04:01:12 - 45 scans - 28 detections (62.0%)	

- Sample B (mcvsmap.exe)

1 Scanned: 2013-03-21 13:29:45 - 44 scans - 0 detections (0.0%)

- Sample C (McUtil.DLL)

1	MicroWorld-eScan:	Trojan.Agent.AZDK
2	nProtect:	Trojan.Agent.AZDK
3	McAfee:	RDN/Generic BackDoor!gt
4	Malwarebytes:	Backdoor.Gulpix
5	Symantec:	WS.Reputation.1
6	Norman:	Agent.APIJH
7	TrendMicro-HouseCall:	TROJ_GEN.RCBCRCJ
8	Avast:	Win32:Gulpix-B [Trj]
9	Kaspersky:	Backdoor.Win32.Gulpix.bo
10	BitDefender:	Trojan.Agent.AZDK
11	Agnitum:	Backdoor.Gulpix!EFaRR6zLtc4
12	Comodo:	UnclassifiedMalware
13	F-Secure:	Trojan.Agent.AZDK
14	DrWeb:	Trojan.Click2.52215
15	VIPRE:	Trojan.Win32.Generic!BT
16	AntiVir:	TR/Agent.azdk.2
17	TrendMicro:	TROJ_GEN.RCBCRCJ
18	McAfee-GW-Edition:	RDN/Generic BackDoor!gt
19	Sophos:	Troj/Agent-AATT
20	Microsoft:	Backdoor:Win32/Plugx.A
21	GData:	Trojan.Agent.AZDK
22	CommTouch:	W32/Backdoor.IYCB-5867
23	Ikarus:	Backdoor.Win32.Gulpix
24	Fortinet:	W32/Gulpix.BO!tr.bdr
25	AVG:	Agent4.AKAP
26	Panda:	Trj/CI.A
27	Scanned: 2013-03-21 13:46:10 - 44 scans - 26 detections (59.0%)	

- Sample D (McUtil.DLL.PPT)

1	Not uploaded to VirusTotal.
---	-----------------------------

- Sample E

1	Not uploaded to VirusTotal.
---	-----------------------------

- Sample F (UAC.TMP)

1	Panda:	Suspicious file
2	Scanned: 2012-09-20 02:33:55 - 43 scans - 1 detections (2.0%)	