

1. General function and functionality of the malware

The malware executes in a command shell, it begins by checking to see if the executing file contains the MZP file extension, and then continues to access the Windows Registry checking to see if a few local registry values exists. The malware will then try to search and display information regarding the executing host (such as a server) including but not limited to the password, port number, ICQ number, and E-mail address.

2. Behavioral patterns of malware

The main portion of the malware gets the host information and then displays back on the command shell next to a predefined text describing the information. This pattern repeats four times until the last piece of server information, E-Mail Address, is displayed. For example:

```
004041DE CALL malware_.0040400C ; Function retrieving server information
004041E3 MOV EAX, DOWORD PRT SS:[EBP-14]
004041E6 PUSH EAX
004041E7 MOV EAX,DWORD PRT DS:[4050E0]
004041EC MOV EDX,malware_.0040432C ; "Password for server : "
004041F1 CALL malware_.00403100 ; Check to see if string is empty
004041F6 POP EDX
004041F7 CALL malware_.00403100 ; Check to see if the popped value is empty
004041FC CALL malware_.00403C53 ; Display text on Cmd prompt
...
0040420B CALL malware_.0040400C ; Retrieve the next server information
...
00404219 MOV EDX,malware_.00404350 ; "Portnumber to server : "
0040421W CALL malware_.00403100 ; Check to see if string is empty
00404223 POP EDX
00404224 CALL malware_.00403100 ; Check to see if the popped value is empty
00404229 CALL malware_.00403C53 ; Display text on Cmd prompt
...
```

3. Local system interaction

The malware uses functions from Windows imported link libraries: kernel32.dll, user32.dll, advapi32.dll, oleaut32.dll. The malware uses critical section objects for mutual exclusion synchronization in the thread of a single process.

```
0040181E CALL <JMP.&kernel32.InitializeCriticalSection> ; Allocate the memory used by
a critical section
00401823 CMP BYTE PTR DS:[406031],0
0040182A JE SHORT malware_.00401836
0040182C PUSH malware_.00406420
```

00401831 CALL <JMP.&kernel32.EnterCriticalSection> ; Waits for ownership of the specified critical section object.

While taking ownership of the specified critical section, the malware allocates a specific number of bytes from the local heap.

00401854 PUSH OFF8
00401859 PUSH 0
0040185B CALL <JMP.&kernel32.LocalAlloc> ; Allocates the specified number of bytes from the heap.

The malware looks for server values in an internal Kernel, but the malware goes into an error routine when the kernel returns nothing.

00404178 CALL malware_.00402560
0040417D CALL malware_.0040266C ; Calls an internal Kernel and then returns value
00404182 TEST EAX,EAX ; EAX returns 00000000
00404184 JLE malware_.0040429D ; Jumps to this error routine returning the word "<ERROR>"
...
0040266C PUSH EBP
...
00402686 LEA EDX,DWORD PTR SS:[EBP-4] ; Stack points to an internal kernel function call ntdll.KiFastSystemCallRet

4. Files and registry keys created, modified and accessed

The malware will look for a registry key under the HKEY_CURRENT_USER name "SOFTWARE\Borland\Delphi\RTL", and then in a later routine the malware looks for the "Software\Borland\Locales" registry key. If that registry key does not exist, then the malware will look for another key name "Software\Borland\Delphi\Locales". After both of these registry keys were not found, the malware tries to load itself as data file with an .ENU, and then an .EN extension file.

; Accessing the "SOFTWARE\Borland\Delphi\RTL" Registry Key.
00402774 PUSH malware_.004027F4 ; "SOFTWARE\Borland\Delphi\RTL"
00402779 PUSH 80000002
0040277E CALL <JMP.&advapi32.RegOpenKeyExA> ; Calling the RegOpenKey function from advapi32.dll
00402783 TEST EAX,EAX
00402785 JNZ SHORT malware_.004027D4 ; Jump to the Open Reg Key function

00402787 XOR EAX,EAX
00402789 PUSH EBP
0040278A PUSH malware_.004027CD
0040278F PUSH DWORD PTR FS:[EAX]
00402792 MOV DWORD PTR FS:[EAX],ESP
00402795 MOV DWORD PTR SS:[EBP-C],4
0040279C LEA EAX,DWORD PTR SS:[EBP-C]

```
0040279F PUSH EAX ; Data
004027A0 PUSH 0
004027A6 PUSH 0
004027A8 PUSH malware_.00402810 ; "FPUMaskValue"
004027AD MOV EAX,DWORD PTR SS:[EBP-4]
004027B0 PUSH EAX ; hKey
004027B1 CALL <JMP.&advapi32.RegQueryValueExA> ; Query the registry value
004027B6 XOR EAX,EAX
004027B8 POP EDX
004027B9 POP ECX
004027BA POP ECX
004027BB MOV DWORD PTR FS:[EAX],EDX
004027BE PUSH malware_.004027D4
```

```
004027C3 MOV EAX,DWORD PTR SS:[EBP-4]
004027C6 PUSH EAX ; hKey
004027C7 CALL <JMP.&advapi32.RegCloseKey> ; Close the registry key
004027CC RETN
```

; Accessing the "Software\Borland\Locales", and the "Software\Borland\Delphi\Locales" key.

```
00403398 PUSH malware_.00403524 ; "Software\Borland\Locales"
0040339D PUSH 80000001
004033A2 CALL <JMP.&advapi32.RegOpenKeyExA> ; Calling the RegOpenKey
function from advapi32.dll
004033A7 TEST EAX,EAX
004033A9 JE SHORT malware_.004033C9 ; If key not found, continue to look for a
sub key
```

```
004033AB LEA EAX,DWORD PTR SS:[EBP-8]
004033AE PUSH EAX
004033AF PUSH 0F003F
004033B4 PUSH 0
004033B6 PUSH malware_.00403540 ; "Software\Borland\Delphi\Locales"
004033BB PUSH 80000001
004033C0 CALL <JMP.&advapi32.RegOpenKeyExA> ; Calling the RegOpenKey
function from advapi32.dll
004033C5 TEST EAX,EAX
004033C7 JNZ SHORT malware_.0040343B ; If key not found, jump to the
0040343B function. See 0040343B function in answer 6 because part of the function
accesses local system information.
```

I think the malware tries to replicate itself using file handling functions from kernel32.dll, but it goes into an error routine before a file writing function is called.

```
00403743 CALL <JMP.&kernel32.CreateFileA>
00403748 CMP EAX,-1 ; Compare to 0FFFFFFFh
0040374B JE malware_.00403803 ; If compare result equals to zero, jump to an
error routine, then exit malware
```

5. Network behavior (including hosts, domains and IP addresses accessed)

No Network behavior was found in the malware.

6. Time and local system dependant features

No Time dependant features in malware, but it does get local language information with the "call GetLocaleInfoA" instruction at 004033A9.

```
0040343B MOV EAX,DWORD PTR SS:[EBP-4]
0040343E PUSH EAX
0040343F LEA EAX,DWORD PTR SS:[EBP-11D]
00403445 PUSH EAX
00403446 CALL <JMP.&kernel32.lstrcpy> ; Call the string copy function
0040344B PUSH 5
0040344D LEA EAX,DWORD PTR SS:[EBP-D]
00403450 PUSH EAX ; String data
00403453 CALL <JMP.&kernel32.GetThreadLocale> ; Calling the local thread from
kernel32.dll
00403458 PUSH EAX
00403459 CALL <JMP.&kernel32.GetLocaleInfoA> ; Function call to lookup and
return local system language information.
0040345E XOR ESI,ESI
00403460 CMP BYTE PTR SS:[EBP-11D],0 ; Stack contained value 43 = 'C'
00403467 JE malware_.0040351C ; Value not equal, therefore function not jumped
```

7. Method and means of communication

The malware display its results on the command prompt. Every time after a server information is found/or not found, the malware always go into a display text function. These following instructions are always executed when displaying server information.

```
004041F1 CALL malware_.00403100 ; Check to see if string is empty
004041F6 POP EDX
004041F7 CALL malware_.00403100 ; Check to see if string is empty
004041FC CALL malware_.00403C53 ; Display text on Cmd prompt
```

8. Original infection vector and propagation methodology

This malware could be distributed under different names via Internet newsgroups and e-mails. This malware must be executed in order to infect user's machines.

9. Use of encryption for storage communication

No use of encryption for storage communication in the malware.

10. Use of self modifying or encrypted code

According to PEiD program, the malware used UPX 0.89.6 – 1.02 / 1.05 – 1.24 -> Markus & Laszlo. The first instruction at 0040D280 is the PUSHAD instruction. PUSHAD decrements the stack pointer (ESP) by 32 to hold the eight double word values. The ECX value is 0012FFB0, and the ESP value is 0012FFA4. After highlighting all 4 bytes of the ECX address in the dump view panel (content of ESP), and went to the hardware breakpoint on access. The program went to a POPAD instruction, followed by JMP malware.00404128 at 0040D3CF. Advance execution by single stepping brought the program to 00404128 with a PUSH EBP instruction.

11. Any information concerning development of malware (compiler type, country of origin, author names/handles, etc.)

The malware was created but Sub7, which was identified by the following decoded string, "SUB7 Server Information Stealer v1.0 – PoLYMoRPHiX". An Internet search found Sub7 as the name of a popular backdoor program. The malware was compiled in Borland C, which was also identified based on the decode string, "Portions Copyright © 1983,99 Borland". This malware could be one of the Sub7 distributed backdoor program gathering server information by execution.