LdPinch Report

Feng Zhu (fzhu001@fiu.edu), Jinpeng Wei (weijp@cs.fiu.edu)

1 Malware General Information

Malware Name: LdPinch (named by ThreatExpert)

File size: 641,536 bytes

File type: PE32 executable (GUI) Intel 80386, for MS Windows

MD5: 48e6a5a9ac9f8a6879ca39a4709d8d67

2 Behavior Analysis

When executed, the malware creates a copy of itself at the following location: %Temp%\system32_.exe. %Temp% is a variable that refers to the temporary folder. By default, %Temp% is C:\Documents and Settings\[UserName]\Local Settings\Temp on Windows NT/2000/XP.

Then the malware creates a file at the location %Temp%\svopst_.bin. The content of svopst_.bin is from the Resource Section [2][3] of the header of system32_.exe. After the creation of svopst_.bin, system32_.exe is deleted. It seems that the malware then decrypts svopst_.bin and based on the decrypted content, creates two files: %Temp%\AMB_1.03.exe and %Temp%\network.ico. However, the exact decryption algorithm used by the malware is still unclear.

Next, the malware tries to run AMB_1.03.exe and uses the API CreateProcessW to create a process for it. In the API CreateProcessW, this file will be checked before creating its process. Due to an unknown reason, the header of AMB_1.03.exe is not a valid executable file header, so the checking result is STATUS_INVALID_IMAGE_NOT_MZ (its value is 0xC000012F, which means that this file does not have the correct format: it does not have an initial MZ [1]). Because the checking result of this file is STATUS_INVALID_IMAGE_NOT_MZ, CreateProcessW launches NTVDM to execute this file.

3 Assembly Code Analysis

To thoroughly understand how exactly the LdPinch sample creates AMB_1.03.exe, we use Ollydbg and IDA disassembler to analyze the sample file C:\ldpinch_48e6a5a9ac9f8a6879ca39a4709d8d67.exe. Since we do not have source code for the malware sample, we can only list assembly code here. The relevant code path is shown and annotated below:

```
00406A7C    55              PUSH EBP                        ; malware entry point
00406A7D    8BEC            MOV EBP,ESP
00406A7F    83C4 F0         ADD ESP,-10
00406A82    B8 3C6A4000     MOV EAX,ldpinch_.00406A3C
00406A87    E8 10DCFFFF     CALL ldpinch_.0040469C
......
004067BC    50              PUSH EAX
004067BD    8D55 F0         LEA EDX,DWORD PTR SS:[EBP-10]
004067C0    33C0            XOR EAX,EAX
004067C2    E8 4DBFFFFF     CALL ldpinch_.00402714
004067C7    8B45 F0         MOV EAX,DWORD PTR SS:[EBP-10]
004067CA    E8 F9D1FFFF     CALL ldpinch_.004039C8
004067CF    50              PUSH EAX
004067D0    E8 83DFFFFF     CALL <JMP.&kernel32.CopyFileA>;ExistingFileName =  "C:\ldpinch_48e6a5a9ac9f8a6879ca39a4709d8d67.exe"
004067D0                                                    ;NewFileName =
```

```
004067D0                                                ;"C:\Documents and Settings\[User]\Local Settings\Temp\system32_.exe"
004067D0                                                ; Copy the malware itself to system32_.exe
.....

00405358   6A 00              PUSH 0
0040535A   6A 00              PUSH 0
0040535C   57                 PUSH EDI
0040535D   6A 00              PUSH 0
0040535F   56                 PUSH ESI
00405360   53                 PUSH EBX
00405361   8B45 FC            MOV EAX,DWORD PTR SS:[EBP-4]
00405364   50                 PUSH EAX                     ; FileName=
00405364                                                    ;"C:\Documents and Settings\[User]\Local Settings\Temp\system32_.exe"
00405365   E8 06F4FFFF    CALL <JMP.&kernel32.CreateFileW>  ; Open system32_.exe,
00405365                                                    ; return value is 48 (file handle for system32_.exe), which stores in EAX
0040536A   8BD8               MOV EBX,EAX                  ; 48->EBX
.....

0040540D   6A 00              PUSH 0                       ; MapName = NULL
0040540F   6A 00              PUSH 0                       ; MaximumSizeLow = 0
00405411   6A 00              PUSH 0                       ; MaximumSizeHigh = 0
00405413   6A 04              PUSH 4                       ; Protection = PAGE_READWRITE
00405415   6A 00              PUSH 0                       ; pSecurity = NULL
00405417   57                 PUSH EDI                     ; hFile = 48
00405418   E8 5BF3FFFF        CALL <JMP.&kernel32.CreateFileMappingA>  ; Create a file mapping object for system32_.exe
00405418                                                    ; return value is 40 (the handle of the file mapping object), which is in EAX
0040541D   8943 04            MOV DWORD PTR DS:[EBX+4],EAX
00405420   837B 04 00         CMP DWORD PTR DS:[EBX+4],0
00405424   0F84 C3000000      JE ldpinch_.004054ED
0040542A   6A 00              PUSH 0                       ; MapSize = 0
0040542C   6A 00              PUSH 0                       ; OffsetLow = 0
0040542E   6A 00              PUSH 0                       ; OffsetHigh = 0
00405430   68 1F000F00        PUSH 0F001F                  ; AccessMode = F001F
00405435   8B43 04            MOV EAX,DWORD PTR DS:[EBX+4]
00405438   50                 PUSH EAX                     ; hMapObject is 40
00405439   E8 92F3FFFF        CALL <JMP.&kernel32.MapViewOfFile>   ; Map the file mapping object of system32_.exe to memory
00405439                                                    ; return value is 930000, which is in EAX
00405439                                                    ; 930000 is the starting address of the memory that stores
00405439                                                    ; the content of system32_.exe
0040543E   8BF8               MOV EDI,EAX
.....

0040547E   40                 INC EAX
0040547F   33FF               XOR EDI,EDI
00405481   8D14BF             LEA EDX,DWORD PTR DS:[EDI+EDI*4]
00405484   8B4D F8            MOV ECX,DWORD PTR SS:[EBP-8]
00405487   8B54D1 0C          MOV EDX,DWORD PTR DS:[ECX+EDX*8+C]   ; parse the Section Headers [3] of system32_.exe
0040548B   8B4B 0C            MOV ECX,DWORD PTR DS:[EBX+C]
0040548E   3B91 88000000      CMP EDX,DWORD PTR DS:[ECX+88]        ; try to locate the resource section of system32_.exe
00405494   75 53              JNZ SHORT ldpinch_.004054E9
00405496   837D 0C 00         CMP DWORD PTR SS:[EBP+C],0
0040549A   74 29              JE SHORT ldpinch_.004054C5
0040549C   8B15 60484000      MOV EDX,DWORD PTR DS:[404860]
004054A2   B8 29000000        MOV EAX,29
004054A7   E8 28ECFFFF        CALL ldpinch_.004040D4
004054AC   8943 10            MOV DWORD PTR DS:[EBX+10],EAX
004054AF   BA 29000000        MOV EDX,29
004054B4   8B43 10            MOV EAX,DWORD PTR DS:[EBX+10]
004054B7   E8 64F3FFFF        CALL ldpinch_.00404820
004054BC   8B43 10            MOV EAX,DWORD PTR DS:[EBX+10]
004054BF   C640 14 01         MOV BYTE PTR DS:[EAX+14],1
004054C3   EB 1F              JMP SHORT ldpinch_.004054E4
004054C5   8D04BF             LEA EAX,DWORD PTR DS:[EDI+EDI*4]
004054C8   8B55 F8            MOV EDX,DWORD PTR SS:[EBP-8]
004054CB   8B4CC2 0C          MOV ECX,DWORD PTR DS:[EDX+EAX*8+C]   ; ECX=7200, the offset of the resource section
004054CB                                                           ; to the beginning of system32_.exe
```

```
004054CF    8D04BF          LEA EAX,DWORD PTR DS:[EDI+EDI*4]
004054D2    8B55 F8         MOV EDX,DWORD PTR SS:[EBP-8]
004054D5    8B54C2 14       MOV EDX,DWORD PTR DS:[EDX+EAX*8+14]  ; EDX=D000, the relative virtual address of the resource section
004054D9    8B43 08         MOV EAX,DWORD PTR DS:[EBX+8]
004054DC    E8 77F7FFFF     CALL ldpinch_.00404C58
004054E1    8943 10         MOV DWORD PTR DS:[EBX+10],EAX
004054E4    8975 FC         MOV DWORD PTR SS:[EBP-4],ESI
004054E7    EB 04           JMP SHORT ldpinch_.004054ED
004054E9    47              INC EDI
004054EA    48              DEC EAX
004054EB    75 94           JNZ SHORT ldpinch_.00405481              ; a loop of parsing the Section Headers
......

00405358    6A 00           PUSH 0
0040535A    6A 00           PUSH 0
0040535C    57              PUSH EDI
0040535D    6A 00           PUSH 0
0040535F    56              PUSH ESI
00405360    53              PUSH EBX
00405361    8B45 FC         MOV EAX,DWORD PTR SS:[EBP-4]
00405364    50              PUSH EAX                         ; FileName=
00405364                                                     ;"C:\Documents and Settings\[User]\Local Settings\Temp\svopst_.bin"
00405365    E8 06F4FFFF     CALL <JMP.&kernel32.CreateFileW>  ; Create svopst_.bin,
00405365                                                     ; return value is 4c (file handle for svopst_.bin), which stores in EAX
0040536A    8BD8            MOV EBX,EAX                      ; 4c->EBX
......

00405B0C    6A 00           PUSH 0                           ; pOverlapped = NULL
00405B0E    8D45 F0         LEA EAX,DWORD PTR SS:[EBP-10]
00405B11    50              PUSH EAX
00405B12    8B45 F4         MOV EAX,DWORD PTR SS:[EBP-C]
00405B15    50              PUSH EAX                         ; nBytesToWrite = 8D751, length of the "RELOC" resource section
00405B16    8B45 F8         MOV EAX,DWORD PTR SS:[EBP-8]
00405B19    50              PUSH EAX                         ; Buffer = 937430, starting address of the "RELOC" resource section
00405B1A    8B45 EC         MOV EAX,DWORD PTR SS:[EBP-14]
00405B1D    50              PUSH EAX                         ; hFile is 4c (handle of svopst_.bin)
00405B1E    E8 F5ECFFFF     CALL <JMP.&kernel32.WriteFile>   ; Write the content of the "RELOC" resource section of system32_.exe
00405B1E                                                     ; to svopst_.bin. So the length of svopst_.bin is 8D751 bytes.
......

004068D7    50              PUSH EAX                 ; FileName="C:\Documents and Settings\[User]\Local Settings\Temp\system32_.exe"
004068D8    E8 A3DEFFFF     CALL <JMP.&kernel32.DeleteFileA>  ; Delete system32_.exe
......

00402C87    6A 00           PUSH 0
00402C89    68 80000000     PUSH 80
00402C8E    51              PUSH ECX
00402C8F    6A 00           PUSH 0
00402C91    52              PUSH EDX
00402C92    50              PUSH EAX
00402C93    8D43 48         LEA EAX,DWORD PTR DS:[EBX+48]
00402C96    50              PUSH EAX             ; FileName= "C:\Documents and Settings\[User]\Local Settings\Temp\svopst_.bin"
00402C97    E8 9CE3FFFF     CALL <JMP.&kernel32.CreateFileA>  ; Open svopst_.bin
00402C97                                                      ; return value is 48 (handle for svopst_.bin)
......

00402A5E    6A 00           PUSH 0
00402A60    8D45 FC         LEA EAX,DWORD PTR SS:[EBP-4]
00402A63    50              PUSH EAX
00402A64    8B43 08         MOV EAX,DWORD PTR DS:[EBX+8]
00402A67    F7EE            IMUL ESI
00402A69    50              PUSH EAX                         ; EAX is 8D751, the length to read
00402A6A    57              PUSH EDI                         ; EDI is 9E0718, starting address of the buffer
00402A6B    8B03            MOV EAX,DWORD PTR DS:[EBX]
00402A6D    50              PUSH EAX                         ; EAX is 48 (handle for svopst_.bin)
00402A6E    FF55 0C         CALL DWORD PTR SS:[EBP+C]        ; read 8D751 bytes (the lengh of svopst_.bin) from svopst_.bin
```

```
00402A6E                                                ; to the buffer starting from 9E0718
......

00402AF0   55              PUSH EBP
00402AF1   8BEC            MOV EBP,ESP
00402AF3   53              PUSH EBX
00402AF4   8B5D 08         MOV EBX,DWORD PTR SS:[EBP+8]
00402AF7   53              PUSH EBX
00402AF8   68 B2D70000     PUSH 0D7B2
00402AFD   68 3C2A4000     PUSH <JMP.&kernel32.WriteFile>
00402B02   6A 65           PUSH 65
00402B04   E8 3BFFFFFF     CALL ldpinch_.00402A44          ; presumably decrypt the content in the buffer starting from 9E0718
......

00402A5E   6A 00           PUSH 0
00402A60   8D45 FC         LEA EAX,DWORD PTR SS:[EBP-4]
00402A63   50              PUSH EAX
00402A64   8B43 08         MOV EAX,DWORD PTR DS:[EBX+8]
00402A67   F7EE            IMUL ESI
00402A69   50              PUSH EAX                     ; EAX is 8D751, the length to write
00402A6A   57              PUSH EDI                     ; EDI is 9E0718, starting address of the buffer
00402A6A                                                ; now the buffer stores the presumably decrypted content
00402A6B   8B03            MOV EAX,DWORD PTR DS:[EBX]
00402A6D   50              PUSH EAX                     ; EAX is 48 (handle for svopst_.bin)
00402A6E   FF55 0C         CALL DWORD PTR SS:[EBP+C]    ; write 8D751 bytes from the buffer to svopst_.bin
......

004064CB   BA C8664000     MOV EDX,ldpinch_.004066C8    ; ASCII "DF6C52F0B8098A48D040D5C099704166.dat"
004064D0   E8 9FD3FFFF     CALL ldpinch_.00403874
004064D5   8B45 C0         MOV EAX,DWORD PTR SS:[EBP-40]
004064D8   E8 EBD4FFFF     CALL ldpinch_.004039C8
004064DD   50              PUSH EAX                     ; FileName=" C:\Documents and Settings\[User]\Local Settings\Temp\
004064DD                                                ; DF6C52F0B8098A48D040D5C099704166.dat"
004064DE   E8 7DE2FFFF     CALL <JMP.&kernel32.CreateFileA>  ; Create DF6C52F0B8098A48D040D5C099704166.dat
004064DE                                                ; return value is 40 (handle for DF6C52F0B8098A48D040D5C099704166.dat)
......

00406548   52              PUSH EDX                     ; BytesToRead=85A00
00406549   8BF3            MOV ESI,EBX
0040654B   56              PUSH ESI                     ; Buffer=00930000
0040654C   57              PUSH EDI                     ; hFile= 48 (handle for svopost_.bin)
0040654D   E8 8EE2FFFF     CALL <JMP.&kernel32.ReadFile>  ; Read 85A00 bytes from svopost_.bin to the buffer 930000
00406552   6A 00           PUSH 0
00406554   8D45 E0         LEA EAX,DWORD PTR SS:[EBP-20]
00406557   50              PUSH EAX
00406558   8B45 E4         MOV EAX,DWORD PTR SS:[EBP-1C]
0040655B   50              PUSH EAX                     ; nBytesToWrite =85A00
0040655C   56              PUSH ESI                     ; Buffer= 00930000
0040655D   8B45 F0         MOV EAX,DWORD PTR SS:[EBP-10]
00406560   50              PUSH EAX                     ; hFile=40 (handle for DF6C52F0B8098A48D040D5C099704166.dat)
00406561   E8 B2E2FFFF     CALL <JMP.&kernel32.WriteFile>  ; Write 85A00 bytes from the buffer to
00406561                                                ; DF6C52F0B8098A48D040D5C099704166.dat. The content in the buffer
00406561                                                ; is from svopost_.bin
......

004061D5   50              PUSH EAX
004061D6   8B45 FC         MOV EAX,DWORD PTR SS:[EBP-4]
004061D9   E8 EAD7FFFF     CALL ldpinch_.004039C8
004061DE   50              PUSH EAX
004061DF   E8 74E5FFFF     CALL <JMP.&kernel32.CopyFileA>; ExistingFileName="C:\Documents and Settings\[User]\
004061DF                                                ; Local Settings\Temp\DF6C52F0B8098A48D040D5C099704166.dat"
004061DF                                                ; NewFileName=" C:\Documents and Settings\[User]\Local Settings\Temp\
004061DF                                                ; AMB_1.03.exe": copy the DAT file to AMB_1.03.exe
......
```

4 Conclusion

The above analysis suggests that the malware hides other malicious executables (in an encrypted form) in its resource section, and during execution it decrypts the embedded executables and drops them onto the disk, before executing them. Our analysis on another LdPinch sample (MD5: f19e796a9cf91356fd0c8e7a a0a516ff) confirms this observation because the decrypted file is a .com file that can be executed. However, it seems that for the malware sample that we analyze in this report, the malware author has made a mistake by packing an invalid executable file: AMB_1.03.exe should be a valid executable file but somehow its format is wrong, which leads to the launching of NTVDM.

5 References

[1] MSDN. http://msdn.microsoft.com/en-us/library/cc704588.aspx

[2] Matt Pietrek. Peering Inside the PE: A Tour of the Win32 Portable Executable File Format. MSDN Magazine, March 1994. Available at http://msdn.microsoft.com/en-us/magazine/ms809762.aspx

[3] Matt Pietrek. An In-Depth Look into the Win32 Portable Executable File Format. MSDN Magazine, February 2002. Available at http://msdn.microsoft.com/en-us/magazine/cc301805.aspx