Matcash Family Report

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

1 Malware General Information

Malware Name: Matcash (named by ThreatExpert)

Malware Type: Adware (Adware is any software package which automatically renders advertisements in order to generate revenue for its author).

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

2 Behavior Analysis

When executed, the malware will download a file from ymq.a[Random Number].wrs.mcboo.com (for example, ymq.a2000352.wrs.mcboo.com). The downloaded file is saved at the following location: %windir%\17PHolmes[Random Number].exe (for example, 17PHolmes2000352.exe). The malware tries to execute the downloaded file so it uses the API CreateProcessW to create a process for it. In the API CreateProcessW, this file will be checked before creating its process. Since this file is an .html file (although it has an .exe extension), the checking result is STATUS_INVALID_IMAGE_NOT_MZ, 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 (NT Virtual DOS Machine) to execute this file.

The downloaded file, 17PHolmes[Random Number].exe, is a HTML page with a redirect link to www.aquirethisname.com. This website claims itself as the best place to find premium domain names and to find the perfect domain name for the visitor's business.

3 Assembly Code Analysis

To better understand how NTVDM starts when the malware tries to execute downloaded files such as 17PHolmes2000352.exe, we use Ollydbg and IDA disassembler to perform an in-depth analysis. We do not have source code for kernel32.dll (which contains the code for CreateProcessW) so we can only list assembly code here. For the functions NtCreateSection, MmCreateSection and MiCreateImageFileMap, since they belong to WRK kernel and we have their source code, we show the C code.

The code path is demonstrated below, and we start from the call to CreateProcessW with the following arguments:

```
lpApplicationName = "C:\WINDOWS\17PHolmes2000352.exe"
lpCommandLine=
"C:\WINDOWS\17PHolmes2000352.exe" 61A847B5BBF72810329B385577FB01F0B3E35B6638993F4661AA4EBD86D67C56389B284534F310"

; The memory address range of kernel32.dll is 7C800000 - 7C8F5FFF

7C802336                mov    edi, edi                ; Entry of CreateProcessW
......
7C80235D                call   CreateProcessInternalW  ; CreateProcessW is a wrapper of CreateProcessInternalW
7C80235D                                               ; Try to create a process for 17PHolmes2000352.exe
......

7C81979C  68 080A0000   PUSH 0A08                      ; Entry of CreateProcessInternalW
……
```

```
7C818FA6   6A 60                PUSH 60
7C818FA8   6A 05                PUSH 5
7C818FAA   8D85 ECF8FFFF        LEA EAX,DWORD PTR SS:[EBP-714]
7C818FB0   50                   PUSH EAX
7C818FB1   8D85 34F8FFFF        LEA EAX,DWORD PTR SS:[EBP-7CC]
7C818FB7   50                   PUSH EAX
7C818FB8   68 A1001000          PUSH 1000A1
7C818FBD   8D85 88F9FFFF        LEA EAX,DWORD PTR SS:[EBP-678]
7C818FC3   50                   PUSH EAX
7C818FC4   8B35 1410807C        MOV ESI,DWORD PTR DS:[<&ntdll.NtOpenFile>   ; Open 17PHolmes2000352.exe and get the file handle
7C818FCA   FFD6                 CALL ESI
......
7C818FE6   FFB5 88F9FFFF        PUSH DWORD PTR SS:[EBP-678]                 ; the file handle of 17PHolmes2000352.exe
7C818FEC   68 00000001          PUSH 1000000
7C818FF1   6A 10                PUSH 10
7C818FF3   53                   PUSH EBX
7C818FF4   53                   PUSH EBX
7C818FF5   68 1F000F00          PUSH 0F001F
7C818FFA   8D85 90F9FFFF        LEA EAX,DWORD PTR SS:[EBP-670]
7C819000   50                   PUSH EAX
7C819001   FF15 7012807C        CALL DWORD PTR DS:[<&ntdll.NtCreateSection>    ; in NtCreateSection 17PHolmes2000352.exe is checked via
                                                                                ; its handle and STATUS_INVALID_IMAGE_NOT_MZ (C000012F) is
                                                                                ; returned

NTSTATUS NtCreateSection (…, __in_opt HANDLE FileHandle){  /* WRK-v1.2\base\ntos\mm\creasect.c */
......
        Status = MmCreateSection (&Section,
                DesiredAccess,
                ObjectAttributes,
                &CapturedSize,
                SectionPageProtection,
                AllocationAttributes,
                FileHandle,
                NULL);
        if (!NT_SUCCESS(Status)) {
......
                return Status;
        }
......
}

NTSTATUS MmCreateSection(…, __in_opt HANDLE FileHandle, __in_opt PFILE_OBJECT FileObject) { /* WRK-v1.2\base\ntos\mm\creasect.c */

......

        Status = ObReferenceObjectByHandle(FileHandle, …, (PVOID *)&File, NULL);
……
        if (AllocationAttributes & SEC_IMAGE) {

                Status = MiCreateImageFileMap (File, &Segment);

        }
......
        if (!NT_SUCCESS(Status)) {
......
                return Status;
        }
}

NTSTATUS MiCreateImageFileMap (IN PFILE_OBJECT File, OUT PSEGMENT *Segment) /* WRK-v1.2\base\ntos\mm\creasect.c */
{
......
PageFrameNumber = MiGetPageForHeader (TRUE);
......
Base = MiCopyHeaderIfResident (File, PageFrameNumber);
......
```

```
DosHeader = (PIMAGE_DOS_HEADER) Base;
   //
   // Check to determine if this is an NT image (PE format) or
   // a DOS image, Win-16 image, or OS/2 image.  If the image is
   // not NT format, return an error indicating which image it
   // appears to be.
   //

if (DosHeader->e_magic != IMAGE_DOS_SIGNATURE) {

      Status = STATUS_INVALID_IMAGE_NOT_MZ;
      goto BadPeImageSegment;
}
......
BadPeImageSegment:
......
    return Status;
}
```

```
                                                                    ; return from NtCreateSection
7C819007   8BF8              MOV EDI,EAX                            ; EAX is C000012F (IMAGE_DOS_SIGNATURE); C000012F->EDI
......
7C81904B   0F8C E5FA0000     JL kernel32.7C828B36
……
7C828B36   B8 2F0100C0       MOV EAX,C000012F
7C828B3B   3BF8              CMP EDI,EAX                            ; Compare the return value of NtCreateSection with C000012F
7C828B3D  0F85 6705FFFF      JNZ kernel32.7C8190AA                  ; Since EDI==EAX, do not jump
7C828B43   50                PUSH EAX
7C828B44   8D85 48F9FFFF     LEA EAX,DWORD PTR SS:[EBP-6B8]
7C828B4A   50                PUSH EAX                               ; "C:\WINDOWS\17PHolmes2000352.exe"
7C828B4B   E8 1F000000       CALL kernel32.7C828B6F                 ; call BaseIsDosApplication
7C828B50   85C0              TEST EAX,EAX                           ; EAX is 1, meaning the malware is recognized as a DOS
                                                                    ; application
......
7C842B7F   50                PUSH EAX
7C842B80   8D85 28F9FFFF     LEA EAX,DWORD PTR SS:[EBP-6D8]
7C842B86   50                PUSH EAX
7C842B87   FFB5 B0F8FFFF     PUSH DWORD PTR SS:[EBP-750]
7C842B8D   E8 0C650200       CALL kernel32.7C86909E                 ; call BaseCreateVDMEnvironment
7C842B8D                                                            ; create a new environment for VDM
......
7C818FA6   6A 60             PUSH 60
7C818FA8   6A 05             PUSH 5
7C818FAA   8D85 ECF8FFFF     LEA EAX,DWORD PTR SS:[EBP-714]
7C818FB0   50                PUSH EAX
7C818FB1   8D85 34F8FFFF     LEA EAX,DWORD PTR SS:[EBP-7CC]
7C818FB7   50                PUSH EAX
7C818FB8   68 A1001000       PUSH 1000A1
7C818FBD   8D85 88F9FFFF     LEA EAX,DWORD PTR SS:[EBP-678]
7C818FC3   50                PUSH EAX
7C818FC4   8B35 1410807C     MOV ESI,DWORD PTR DS:[<&ntdll.NtOpenFile>  ; Open ntvdm.exe for execute access
7C818FCA   FFD6              CALL ESI
......
7C81927B   66:8B85 ECF7FFFF  MOV AX,WORD PTR SS:[EBP-814]
7C819282   66:2D 4C01        SUB AX,14C
7C819286   66:F7D8           NEG AX
7C819289   1BC0              SBB EAX,EAX
7C81928B   F7D0              NOT EAX
7C81928D   2385 74F9FFFF     AND EAX,DWORD PTR SS:[EBP-68C]
7C819293   8985 00F7FFFF     MOV DWORD PTR SS:[EBP-900],EAX
7C819299   FFB5 1CF7FFFF     PUSH DWORD PTR SS:[EBP-8E4]
7C81929F   53                PUSH EBX
7C8192A0   FFB5 30F7FFFF     PUSH DWORD PTR SS:[EBP-8D0]
7C8192A6   FFB5 90F9FFFF     PUSH DWORD PTR SS:[EBP-670]
7C8192AC   FFB5 FCF7FFFF     PUSH DWORD PTR SS:[EBP-804]
7C8192B2   83CE FF           OR ESI,FFFFFFFF
```

```
7C8192B5  56                PUSH ESI
7C8192B6  FFB5 C8F7FFFF     PUSH DWORD PTR SS:[EBP-838]
7C8192BC  68 FF0F1F00       PUSH 1F0FFF
7C8192C1  8D85 94F9FFFF     LEA EAX,DWORD PTR SS:[EBP-66C]
7C8192C7  50                PUSH EAX
7C8192C8  FF15 5014807C     CALL DWORD PTR DS:[<&ntdll.NtCreateProcess>  ; Create Process for ntvdm.exe
……
```

4 Conclusion

We also analyze the other Matcash samples that trigger NTVDM and find that they work in a similar fashion as the one that we discussed above. In summary, these Matcash samples connect to a remote site (ymq.a[Random Number].wrs.mcboo.com) to download and execute 17PHolmes[Random Number].exe, which leads to the launching of NTVDM as we have discussed in Section 2.

In our opinion, one possible reason why these samples try to execute an .html file is that at the time when they were developed, the downloaded file was indeed an executable file that could automatically deliver advertisement when executed; but the remote site has changed ever since, so the downloaded file becomes an .html file now. However, these Matcash samples still try to execute the downloaded file, blindly assuming that it is the expected one.

A side note of this analysis is: a malware writer can be sloppy because he/she does not bother to verify whether the downloaded file has the right format.

5 References

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