

# Ukraine: Analysis of the new disk-wiping malware (HermeticWiper)

 cluster25.io/2022/02/24/ukraine-analysis-of-the-new-disk-wiping-malware/

February 24, 2022



ESET research  
@ESETresearch

...

Breaking. #ESETResearch discovered a new data wiper malware used in Ukraine today. ESET telemetry shows that it was installed on hundreds of machines in the country. This follows the DDoS attacks against several Ukrainian websites earlier today 1/n

[Traduci il Tweet](#)

APT Cluster25 today February 24, 2022

Very recently a new type of destructive malware named by the security community “**HermeticWiper**” was used to attack organizations and entities in Ukraine shortly before Russia began military operations against the same country. **HermeticWiper** is an executable file signed with a likely stolen certificate issued to **Hermetica Digital Ltd**. It has been reported by researchers at ESET on 2022/02/23



ESET research  
@ESETresearch

...

Breaking. #ESETResearch discovered a new data wiper malware used in Ukraine today. ESET telemetry shows that it was installed on hundreds of machines in the country. This follows the DDoS attacks against several Ukrainian websites earlier today 1/n

[Traduci il Tweet](#)

## Insights

We analyzed the file identified by the following **SHA256** hash :

**1bc44eef75779e3ca1eefb8ff5a64807dbc942b1e4a2672d77b9f6928d292591**

On the basis of some evidences extracted from the analyzed samples, it is possible to hypothesize that the preparation and the development phases of this piece of malware (and may be the military operation that soon followed its spread) had been going on for some months already.

Technically speaking the first actions carried out when executed are aimed at opening its current process using the **GetCurrentProcess API** and at accessing the token with the **OpenProcessToken**.

Immediately after it adjusts the token privileges using the **AdjustTokenPrivileges** to add the **SeShutdownPrivilege** and **SeBackupPrivilege**.

It proceeds by dynamically resolving **Wow64DisableWow64FsRedirection**, **Wow64RevertWow64FsRedirection**, **IsWow64Process** from **kernel32.dll** library using the **GetModuleHandle** and the **GetProcAddress** API's.

OS version is verified through **IsWow64Process** and **VerifyVersionInfoW** API's. Based on the OS version load 1 of the 4 resources as shown following:

```
v5 = VerSetConditionMask(0i64, 2u, 3u);
v6 = VerSetConditionMask(v5, 1u, 3u);
if ( VerifyVersionInfoW(&VersionInformation, 3u, v6) )
{
    if ( v40 )
        v7 = FindResourceW(hModule, L"DRV_X64", L"RCDATA");
    else
        v7 = FindResourceW(hModule, L"DRV_X86", L"RCDATA");
}
else
{
    if ( GetLastError() != 1150 )
        return 0;
    v35 = 1;
    if ( v40 )
        v7 = FindResourceW(hModule, L"DRV_XP_X64", L"RCDATA");
    else
        v7 = FindResourceW(hModule, L"DRV_XP_X86", L"RCDATA");
}
```

The resources contain **ms-compressed** copies of the **empntdrv.sys** from **EaseUS** driver suite used in the next subroutines to access physical drives and getting partition information.

This adds an obfuscation layer to the current wiper since a lot of functionality are accessed through the **DeviceIoControl** API (calling the specific IOCTLs).

After that, the resource is loaded and decompressed. Next, the

**SYSTEM\CurrentControlSet\Control\CrashControl**

registry key is opened to disable the **CrashDump** feature (enabled by default in Windows).

In particular, the DWORD CrashDumpEnabled value is changed from **1** to **0**.

```
loc_402B71:
lea    eax, [ebp+phkResult]
mov    [ebp+phkResult], 0
push   eax           ; phkResult
push   offset SubKey ; "SYSTEM\\CurrentControlSet\\Control\\Cra"...
push   80000002h     ; hKey
call   ds:RegOpenKeyW
test   eax, eax
jnz   short loc_402BB4

push   4             ; cbData
mov    dword ptr [ebp+Data], eax
lea   eax, [ebp+Data]
push   eax           ; lpData
push   4             ; dwType
push   0             ; Reserved
push   offset ValueName ; "CrashDumpEnabled"
push   [ebp+phkResult] ; hKey
call   ds:RegSetValueExW ; Disable CrashDump
push   [ebp+phkResult] ; hKey
call   ds:RegCloseKey
```

Subsequently, a new pipe called `\\\\.\\EPMNTDRV\\` is created and the decompressed driver is saved in the

**C:\Windows\System32\drivers\njdr.sys**

directory (the filename is computed at runtime using randomness).

To ensure the driver execution the process token is modified again to add the **SeLoadDriverPrivilege** permission and a new service is created using the **CreateServiceW** API.

Instead, if the service already exists, the **ChangeServiceConfigW** API is used to force the execution of the service with the **SERVICE\_DEMAND\_START** flag. Finally, the **StartServiceW** API is used to start the service pointing to the utility driver.

After some sleep, the reg key

**SYSTEM\\CurrentControlSet\\services\\**

and the driver file stored on the filesystem are deleted.

```

lea    eax, [ebp+Subkey]
push   eax                ; lpSubKey
push   80000002h          ; hKey
call   ds:RegDeleteKeyW ; delete SYSTEM\\CurrentControlSet\\services\\

loc_402EE8:
mov    edx, [ebp+var_20]
mov    ecx, ebx           ; psz1
call   sub_4023C0
mov    edi, ds>DeleteFileW

loc_402EF8:
push   ebx                ; lpFileName
call   edi ; DeleteFileW
push   ebx                ; pszPath
call   ds:PathFindExtensionW
test   eax, eax
jz     short loc_402F0E

```

Subsequently, the **vss** service (**shadow copies service**) is opened and the **ChangeServiceConfigW** API is used to set the flags **SERVICE\_DISABLED** and **SERVICE\_CONTROL\_STOP**.

Afterwards, the system physical drivers are enumerated multiple times with a counter from **1** to **100**.

```

wprintfW(pszDest, 260, L"\\\\.\\PhysicalDrive%u", a1);
v5 = mw_createpipe_getdevicenum(pszDest, (int)&v26, (int)v25);
v6 = (void *)v5;
if ( v5 != -1 )
{
    if ( !v5 )
        return 0;
    v7 = 9408;
    v8 = GetProcessHeap();
    v3 = (LARGE_INTEGER *)HeapAlloc(v8, 8u, 0x24C0u);
    DeviceIoControl(v6, 0x70050u, 0, 0, v3, 0x24C0u, &BytesReturned, 0);
}

```

For each found physical drive, the **\\.\EPMNDRV\** is called with the appropriate device number using the **DeviceIoControl** API.

Then, for each driver found, the malware starts a bit swapping routine to corrupt the **Master Boot Record** (MBR) for every physical drive.

The corruption is a little different between **FAT** and **NTFS** partitions.

```

if ( lstrcmpA(String1, "NTFS ") )
{
  *(_DWORD *)String1 = *(_DWORD *)(a2 + 54);
  v18 = *(_DWORD *)(a2 + 58);
  v19 = 0;
  if ( StrStrA(String1, "FAT")
      || (v9 = *(_DWORD *)(a2 + 82),
          v18 = *(_DWORD *)(a2 + 86),
          *(_DWORD *)String1 = v9,
          (result = StrStrA(String1, "FAT")) != 0) )
  {
    v11 = *(unsigned __int16 *)(a2 + 22);
    if ( !(_WORD)v11 )
      v11 = *(_DWORD *)(a2 + 36);
    v10 = *(unsigned __int16 *)(a2 + 11);
    sub_401590(
      a4,
      a3,
      a5 + v10 * *(unsigned __int16 *)(a2 + 14),

```

For the **NTFS** partitions, the malware parses the **Master File Table** (MFT) before calling the bit swapping routine.

The bit swapping routine is supported with the usage of cryptographic context API's (like **CryptoAcquireContext**), the generation of random bytes to corrupt the partition and the usage of **DeviceIoControl** with the just-created service to access the physical drives.

Further functionality refers also to the encryption of specific **MFT** fields (**\$bitmap** and **\$logfile**) or **NTFS** streams like **\$DATA**, **\$I30** or **\$INDEX\_ALLOCATION** with some references to "ntuser".

After the corruption subroutines, the malware enumerates also common folders but currently still not totally clear why as the wiping activities are already been operated. Anyway, the folder are the following:

1. My Documents
2. Desktop
3. AppData
4. Windows Event Logs (C:\\Windows\\System32\\winevt\\Logs)

At the end this piece of malware waits for different sleeping threads and initialize a system shutdown destroying the victim system.

## Conclusions

---

**HermeticWiper** is a new type of destructive malware, significantly different from **WhisperGate**, the previous wiper used against targets in Ukraine. Destructive attacks perpetrated through pieces of malware of this type are focused on making essential services and critical infrastructures unavailable.

In this case, **HermeticWiper** probably works in support of Russian military operations.

## Detection

---

### YARA 1

```
rule UNC1222_HermeticWiper_23433_10001 {
  meta:
    date = "2022-02-23"
    description = "Detects HermeticWiper variants by internal strings"
    author = "Cluster25"
    tlp = "white"
    hash1 = "0385eeab00e946a302b24a91dea4187c1210597b8e17cd9e2230450f5ece21da"
    hash2 = "1bc44eef75779e3ca1eefb8ff5a64807dbc942b1e4a2672d77b9f6928d292591"
  strings:
    $ = "tdrv.pdb" fullword ascii
    $ = "\\.\EPMNTDRV\\%u" fullword wide
    $ = "PhysicalDrive%u" fullword wide
    $ = "Hermetica Digital Ltd"
  condition:
    (uint16(0) == 0x5a4d and all of them)
}
```

### YARA 2

```
import "pe"
rule UNC1222_HermeticWiper_23433_10002 {
  meta:
    date = "2022-02-23"
    description = "Detects HermeticWiper variants by internal strings"
    hash1 = "0385eeab00e946a302b24a91dea4187c1210597b8e17cd9e2230450f5ece21da"
    hash2 = "1bc44eef75779e3ca1eefb8ff5a64807dbc942b1e4a2672d77b9f6928d292591"
    tlp = "white"
  strings:
    $p1 = "$INDEX_ALLOCATION" wide
    $p2 = "$I30" wide
    $p3 = "$DATA" wide
    $p4 = "$logfile" wide
    $p5 = "$bitmap" wide
```

```

$s1 = "PhysicalDrive%u" wide
$s2 = "EPMNTDRV" wide
$s3 = "SYSVOL" wide
$s4 = "SYSTEM\\CurrentControlSet\\Control\\CrashControl" wide
$s5 = "CrashDumpEnabled" wide
$s6 = "NTFS" ascii
$s7 = "FAT" ascii
$s8 = "OpenSCManager" ascii
$s9 = "SeBackupPrivilege" wide
$s10 = "SeLoadDriverPrivilege" wide
$s11 = "RCDATA" wide
// LookupPrivilegeValueW routine
$r1 = { 85 35 2C 50 40 00 C7 84 ?? ?? ?? ?? 77 00 6E 00 C7 84 ?? ?? ?? ?? 50 00 72 00
8D 43 04 50 8D 44 24 44 50 6A 00 FF D6 8D 43 10 50 68 A8 55 40 00 6A 00 FF D6 6A 00
6A 00 6A 00 53 C7 03 02 00 00 00 6A 00 }
// AdjustTokenPrivileges routine
$r2 = { C7 43 0C 02 00 00 00 C7 43 18 02 00 00 00 FF 74 24 24 FF 15 28 50 40 00 FF D7
85 C0 75 0F }
// OpenSCManagerW (DatabaseName: "ServicesActive") routine
$r3 = { 68 ?? 3f 00 0f 00 68 ?? 80 55 44 00 33 f6 56 ff 15 24 50 40 00 89 44 24 10 85 C0 75
06 }
// OpenServiceW (ServiceName: "vss") routine
$r4 = { 68 ?? 58 40 00 50 FF 15 20 50 40 00 8B D8 85 DB 75 0C }
// ChangeServiceConfigW routine
$r5 = { 6A 00 6A 00 6A 00 6A 00 6A 00 6A 00 6A 00 6A FF 6A 04 6A 10 53 FF 15 14 50 40
00 85 C0 75 04 }
// CreateThread/CreateEventW and InitializeShutdownW routine
$r6 = { 8B 35 ?? ?? ?? ?? 8D 44 ?? ?? 6A 00 6A 00 50 68 ?? ?? 40 00 6A 00 6A 00 89 7C
?? ?? FF D6 6A 00 6A 00 6A 01 6A 00 89 44 ?? ?? FF 15 ?? ?? ?? ?? 6A 00 6A 00 89 44 ??
?? 8D 44 ?? ?? 50 68 D0 34 40 00 6A 00 6A 00 FF D6 8B 3D D4 ?? ?? ?? 6B D8 85 DB 74
0A }

```

condition:

```

uint16(0)==0x5a4d and pe.imports("lz32.dll") and filesize < 200KB and (2 of ($p*) and (all of
($s*) or (6 of ($s*) and any of ($r*)) or 4 of ($r*)))
}

```

Written by: [Cluster25](#)

Tagged as: [HermeticWiper](#), [Wiper](#), [Ukraine](#), [CyberWar](#), [Malware](#).