Technical Analysis of The Hermetic Wiper Malware Used to Target Ukraine

cloudsek.com/technical-analysis-of-the-hermetic-wiper-malware-used-to-target-ukraine/

Anandeshwar Unnikrishnan March 2, 2022

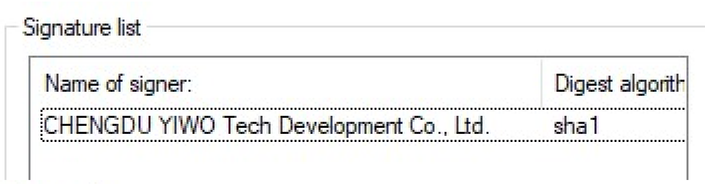
Executive Summary

- 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.

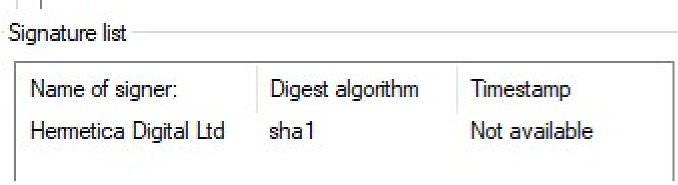
Technical Analysis of Hermetic Wiper Malware

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.



Signed digital certificates of the binaries

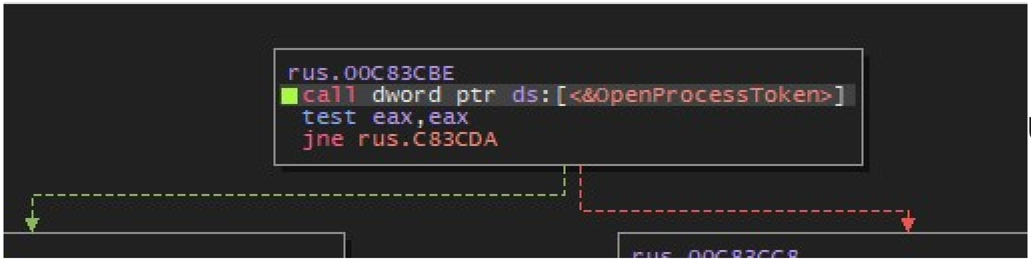


Signed digital certificates of the binaries

Obtaining a Handle to the Current Process Token

- The malware begins by obtaining a handle to the current process' token.

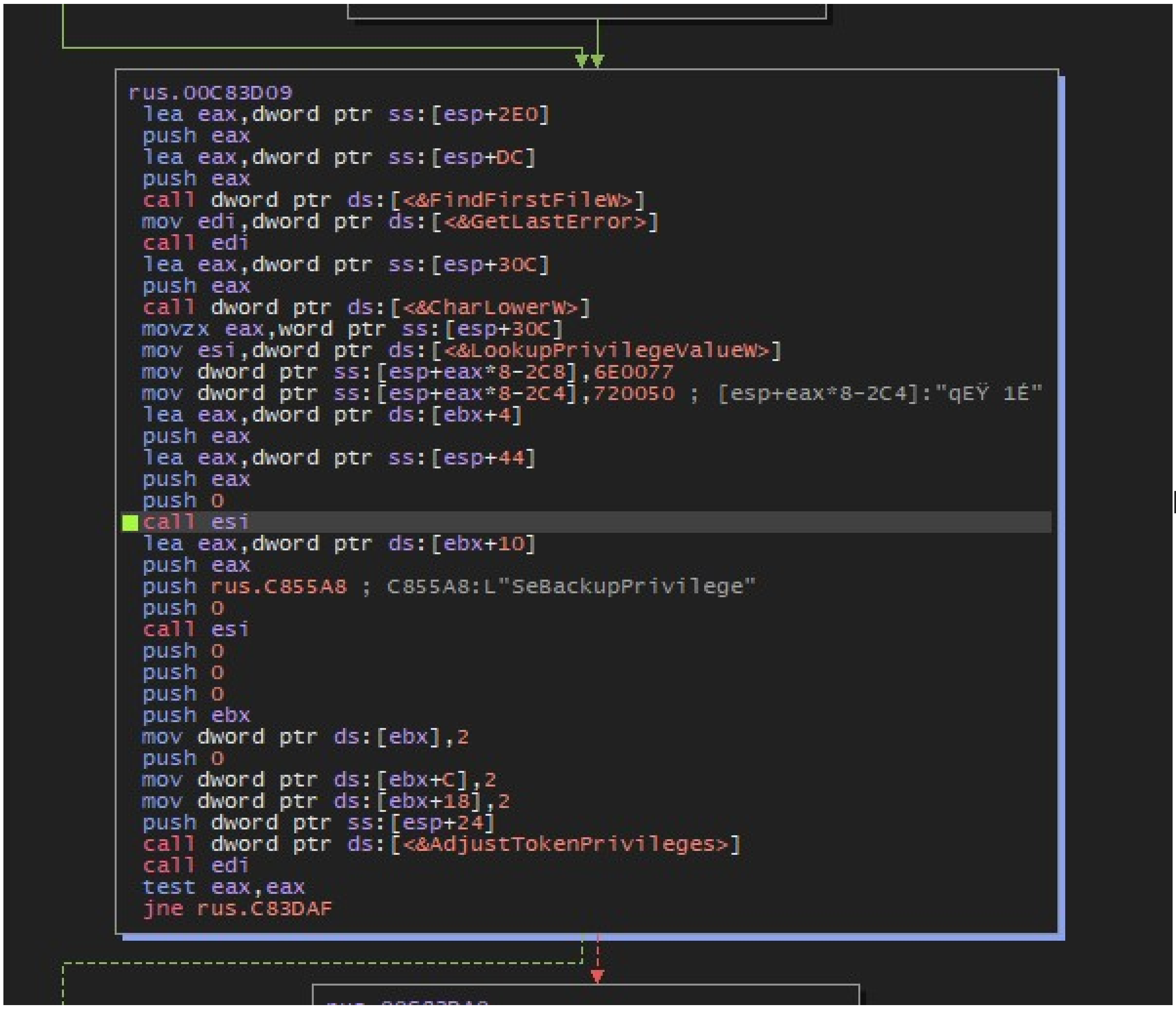
- In Windows, a token is an object that represents the privilege a process holds while running on the system. A full list of privilege constants can be found here.
- The malware uses *OpenProcessToken* API with the *DesiredAccess* parameter set to 0x0028 (TOKEN_QUERY 0x0008 | TOKEN_ADJUST_PRIVILEGES 0x0020).
- This allows the malware to change privileges assigned to the token.



Using OpenProcessToken API to change token privileges

Changing Token Privileges

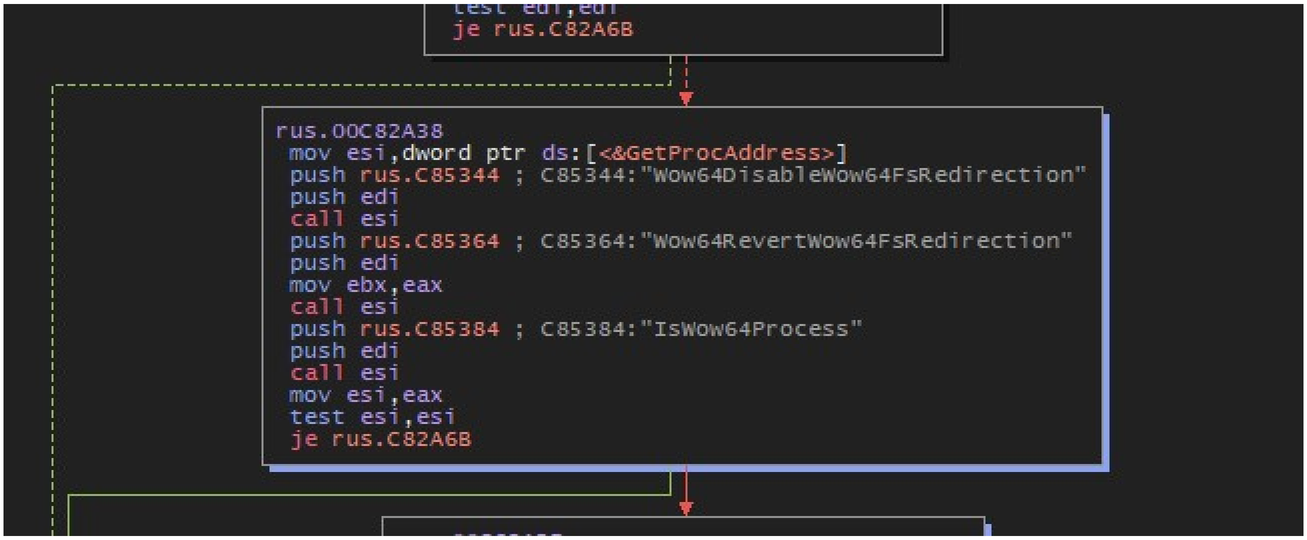
- After obtaining access to the current process' token, with privileges set, the malware uses *LookUpPrivilegeValueW* to make sure the current process is assigned following privileges:
  - SeShutdownPrivilege
  - SeBackupPrivilege
- The *AdjustTokenPrivileges* API is used to adjust the current token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



Process of changing token privileges of the current process

Loading the Payload into the System Memory

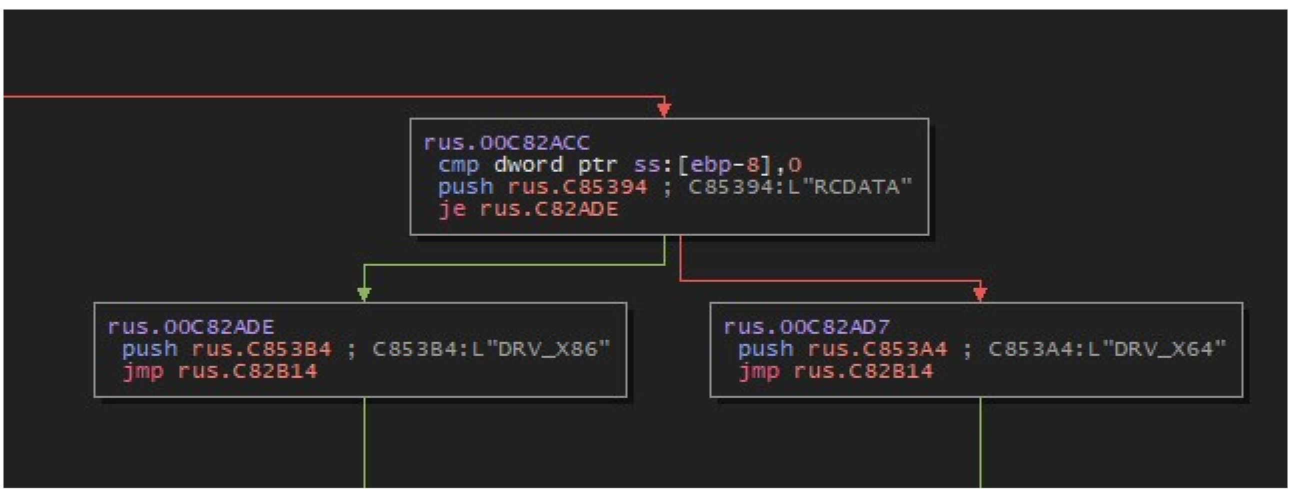
- After granting privileges, the malware dynamically resolves the addresses of following modules and load them into the current process:
  - *Wow64DisableWow64FsRedirection*
  - *Wow64RevertWow64FsRedirection*
  - *IsWow64Process*
- The *Wow64DisableWow64FsRedirection* and *Wow64RevertWow64FsRedirection* 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.



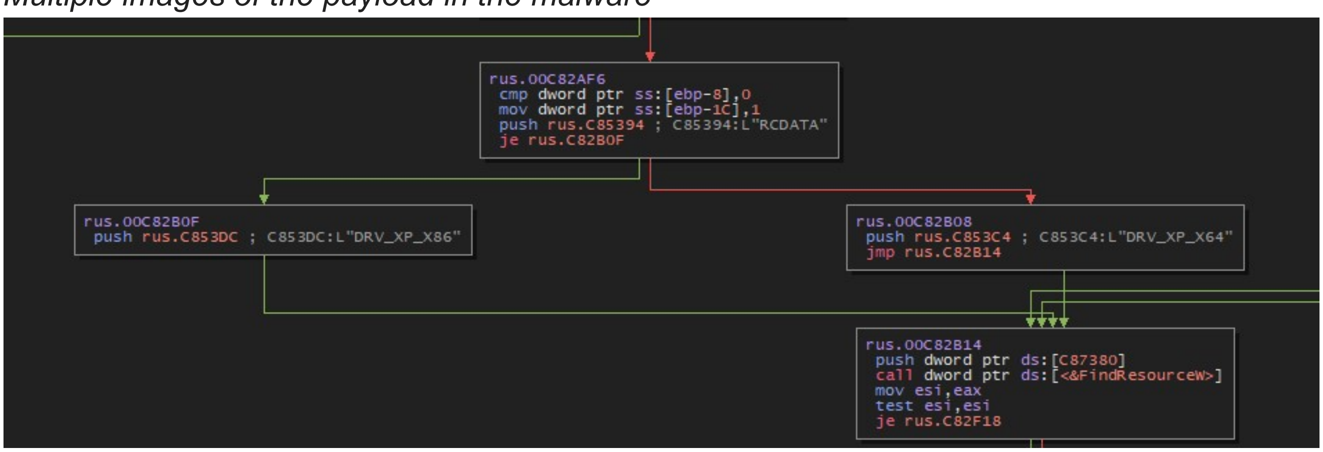
The malware loads the modules into the current process

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

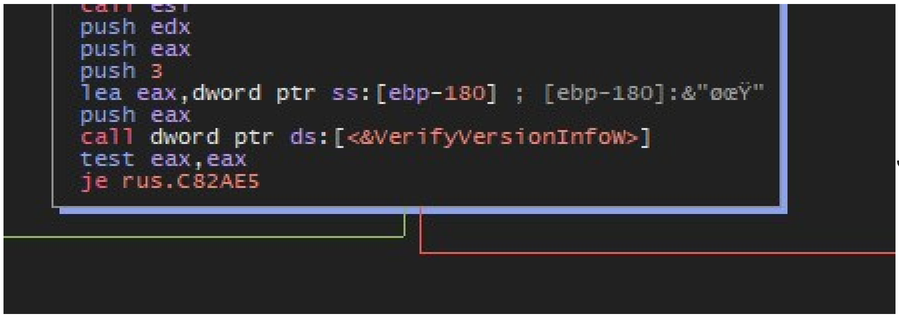


Multiple images of the payload in the malware



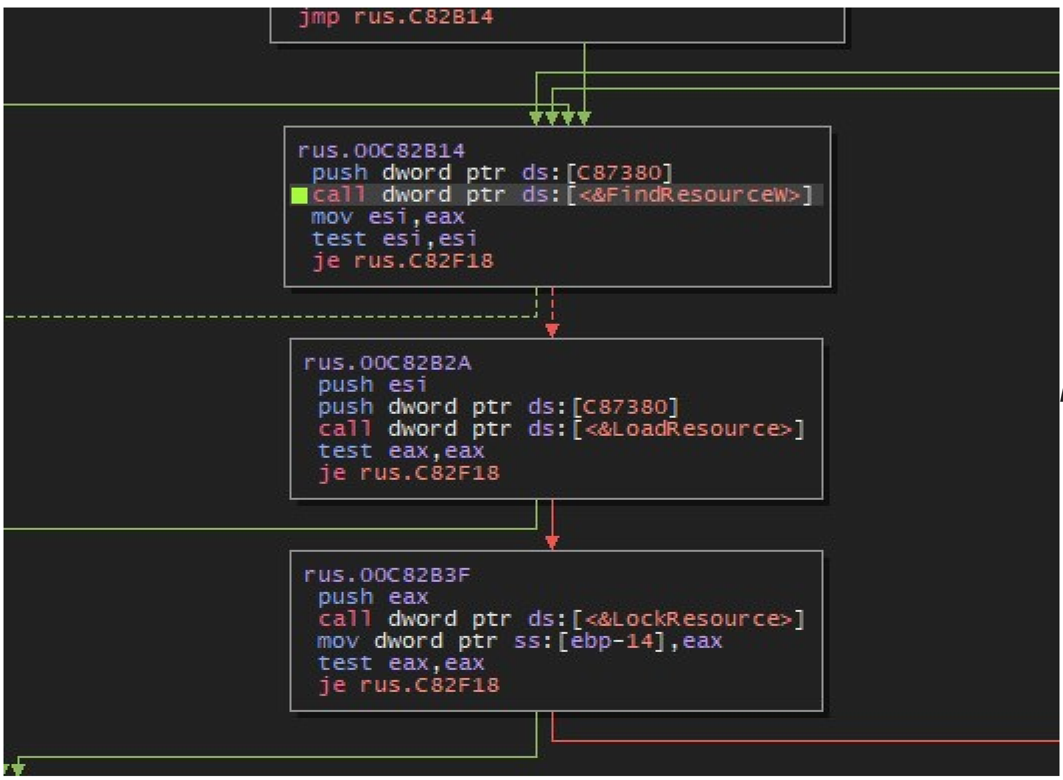
Multiple images of the payload in the malware

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



System version enumeration using VerifyVersionInfoW

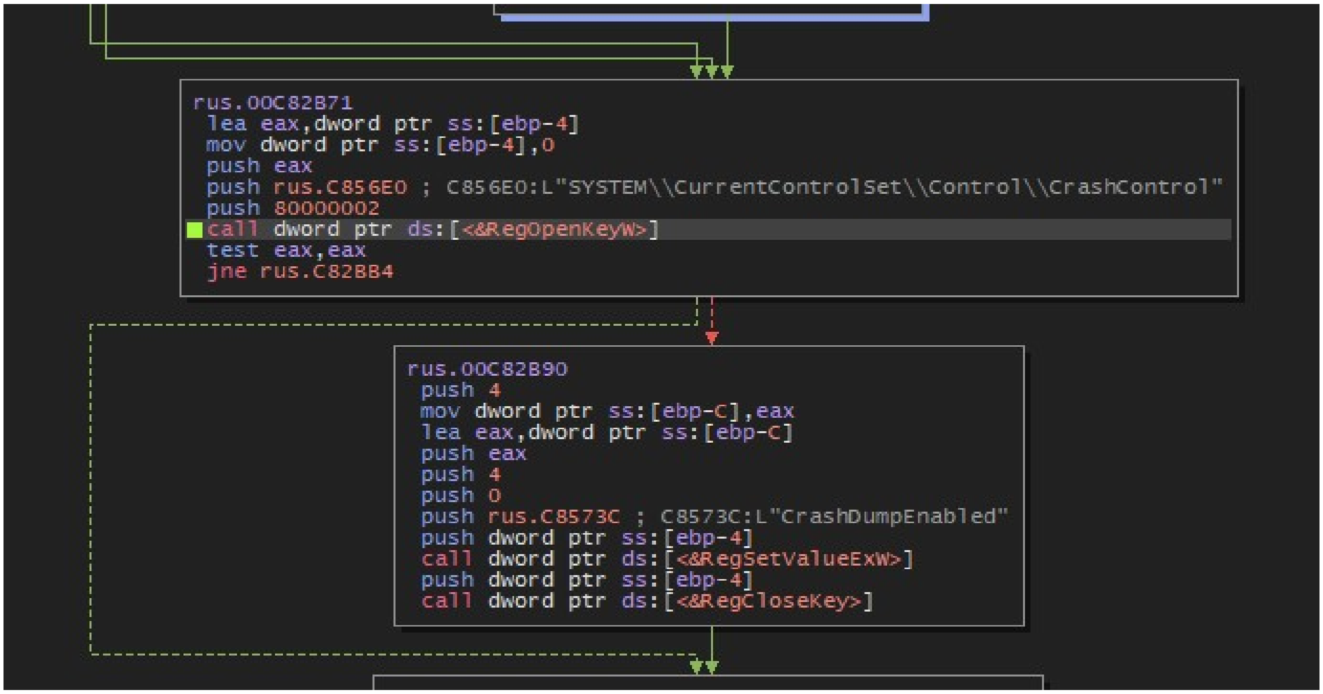
The malware selects the payload to be dropped based on the bitness (32/64) in the resource section of the Portable Executable (PE). After which, the corresponding image is retrieved from the .rsrc section of the malware and loaded into the system memory.



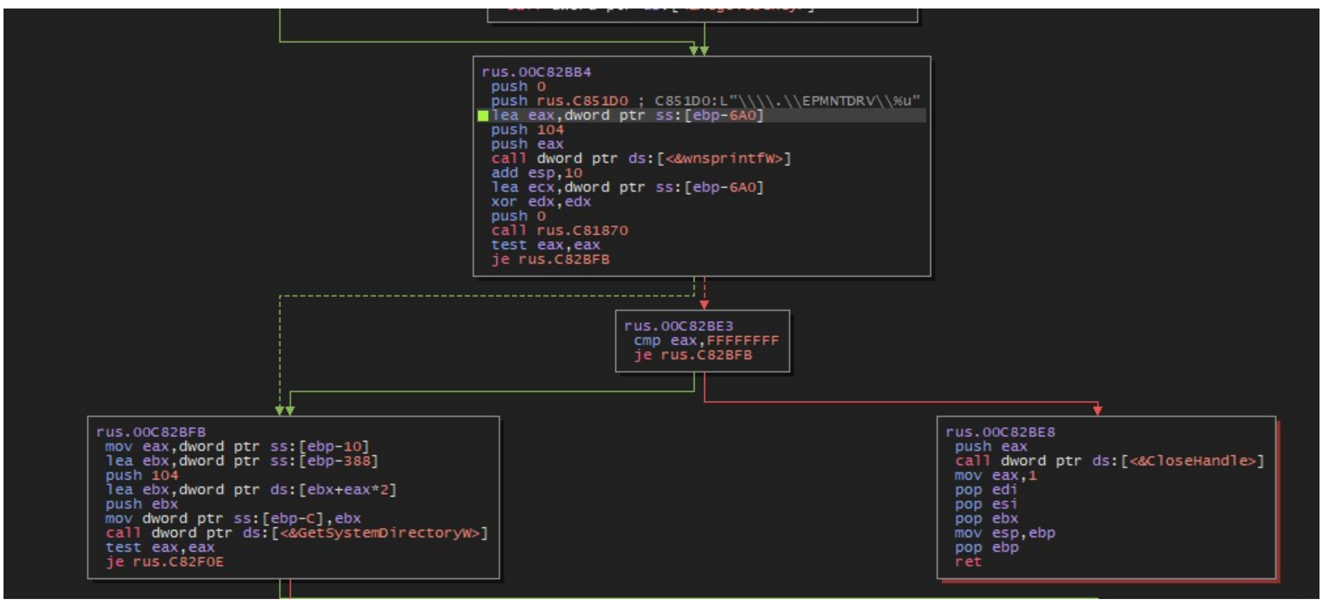
Loading the corresponding payload into the system memory

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.

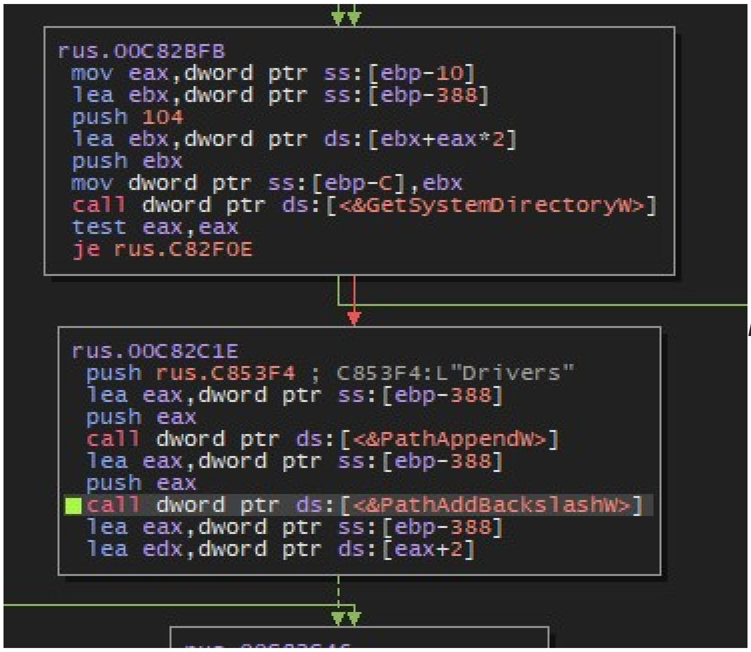


- After disabling the crash dump setting in the registry, the malware prepares to copy an image of the driver, kept in the .rsrc section of the PE, to the target system.
- A pipe \\\\.*EPMNTDRV*\\ is created to perform the transfer of the data.



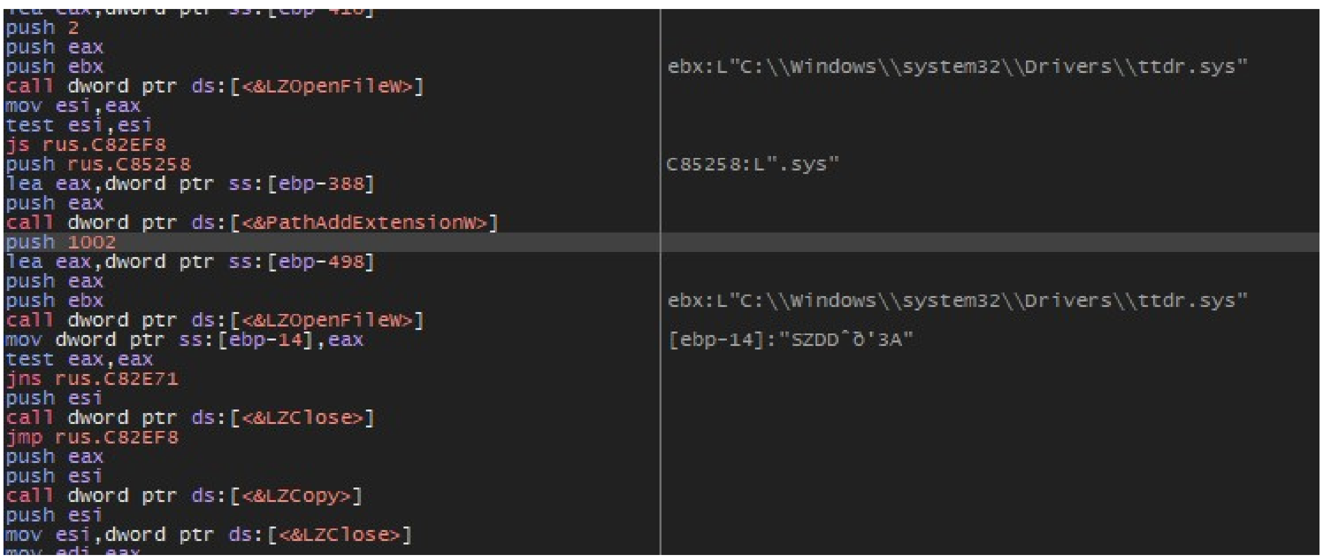
Creating a pipe to transfer data

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



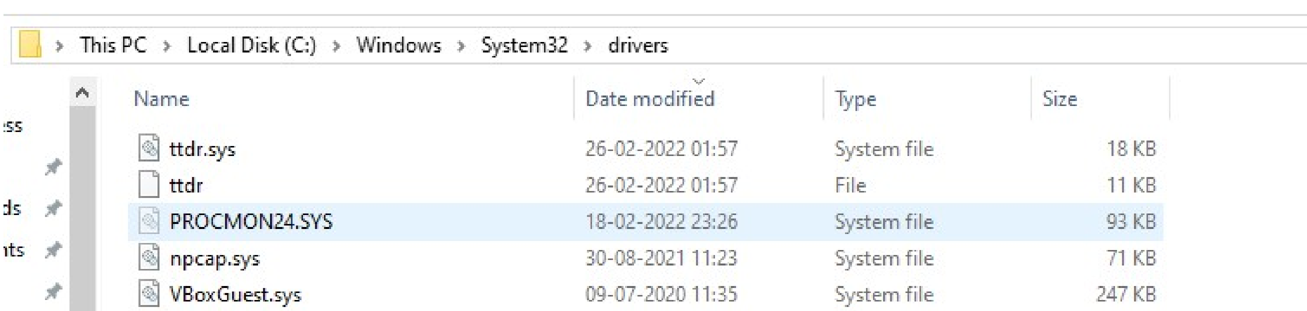
Retrieving the system directory for drivers

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*.



Writing the data and generating the ttdr.sys file

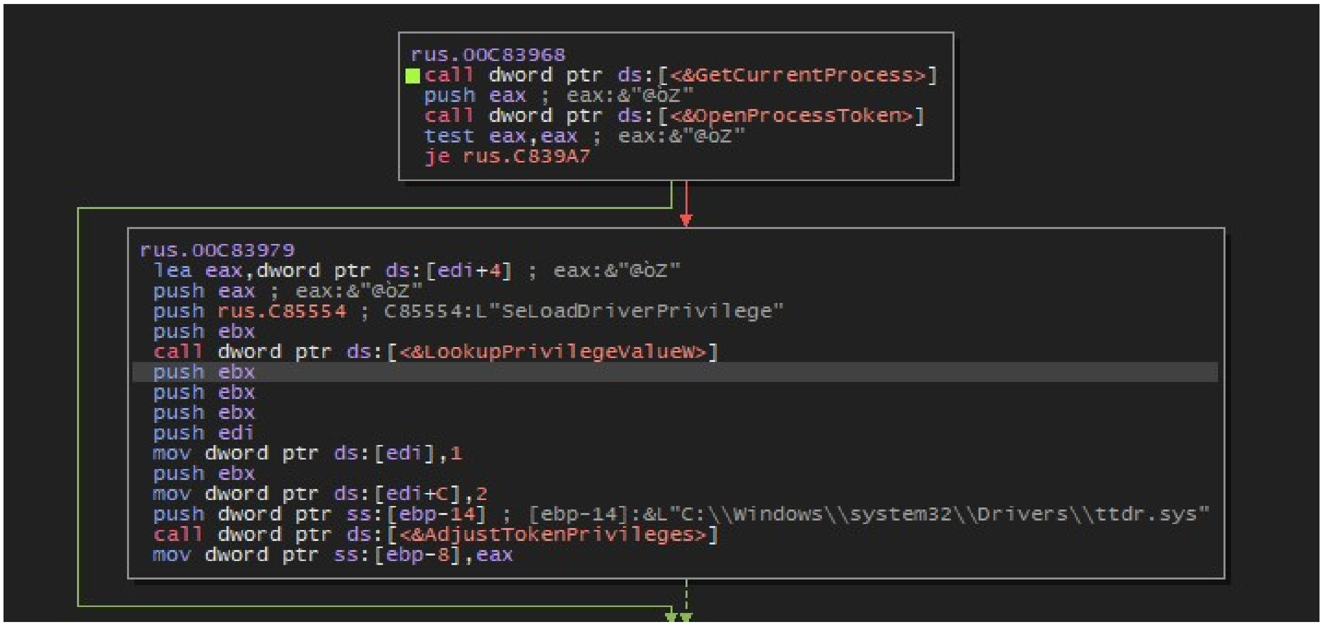
Now the malware has successfully dumped a driver on the target system.



ttdr.sys dumped on the target system

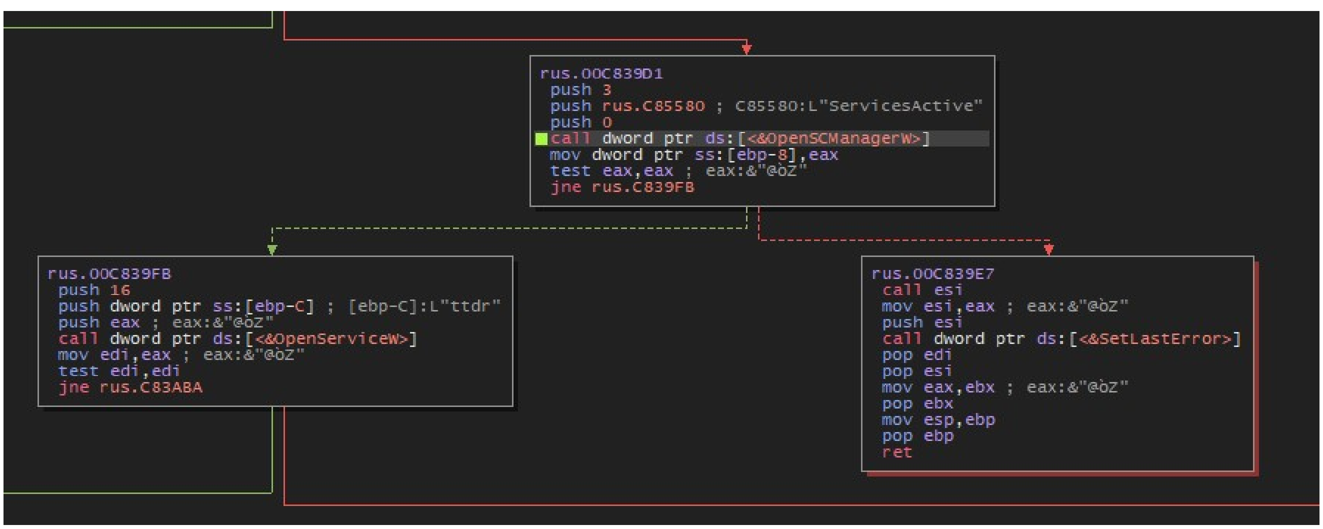
Driver Loading and Service Creation

- 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.



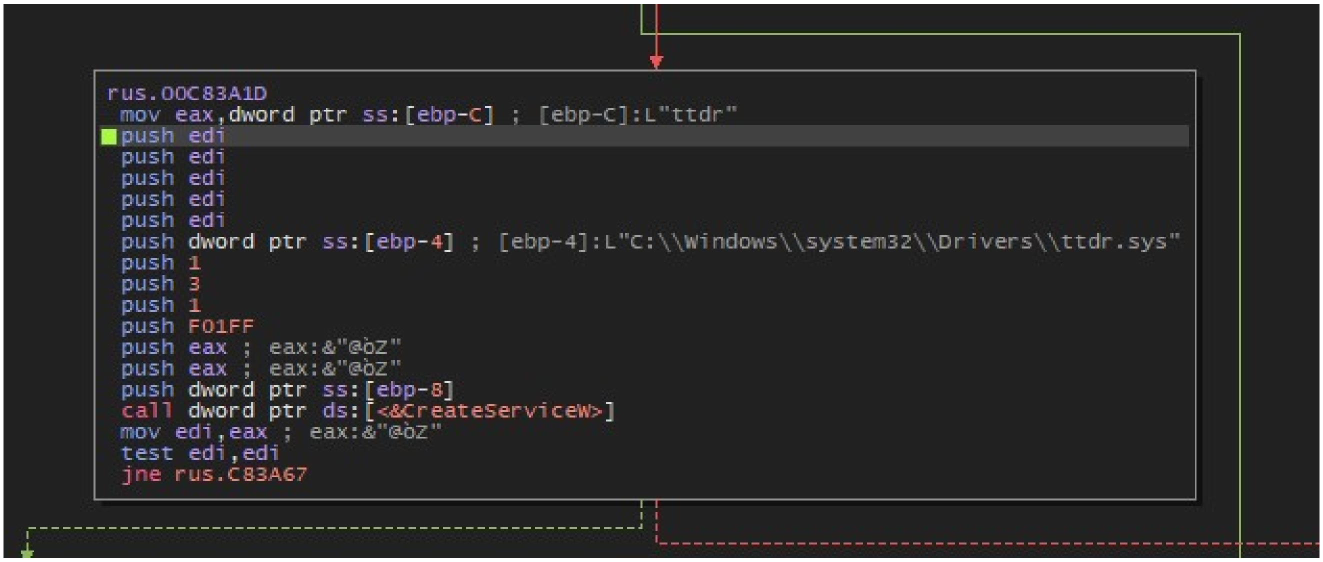
Malware looks for SeLoadDriverPrivilege

- 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.



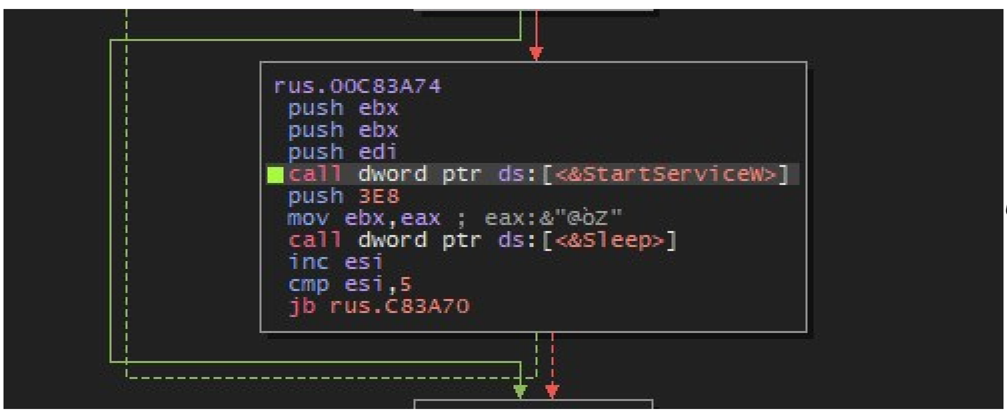
Checking for any active services with the name ttdr

If the service does not exist, *OpenServiceW* API returns *ERROROSERVICE_DOES_NOTEXIST,* and then new service is created via CreateServiceW as shown below



New service is created via CreateServiceW

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



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
```

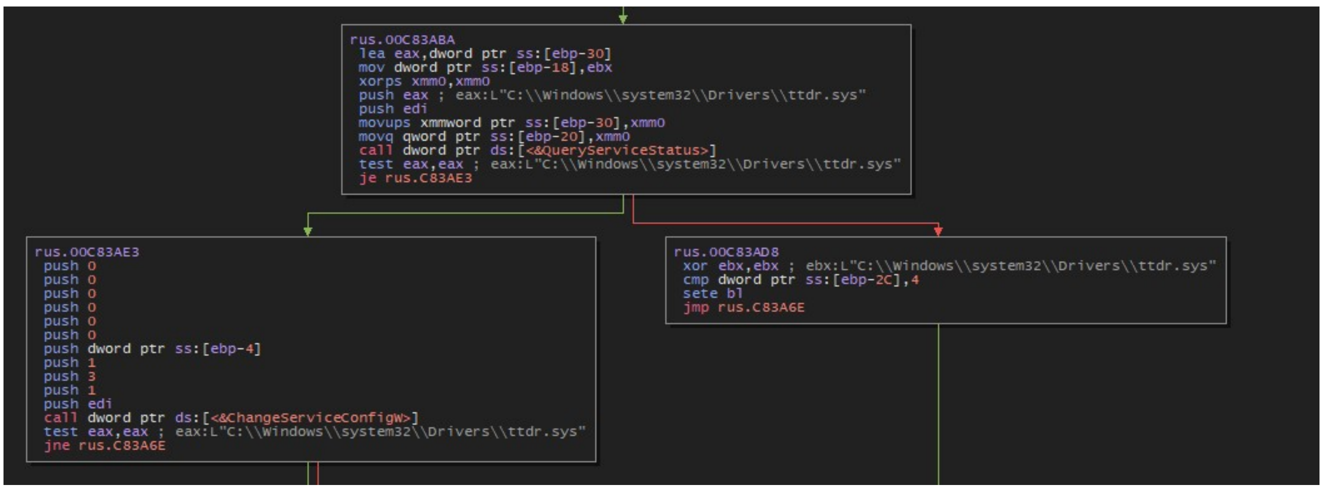
Querying the service control

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

Name	Type	Symbolic Link Target
EPMNTDRV	SymbolicLink	\Device\EPMNTDRV

Symbolic link created for IO communications

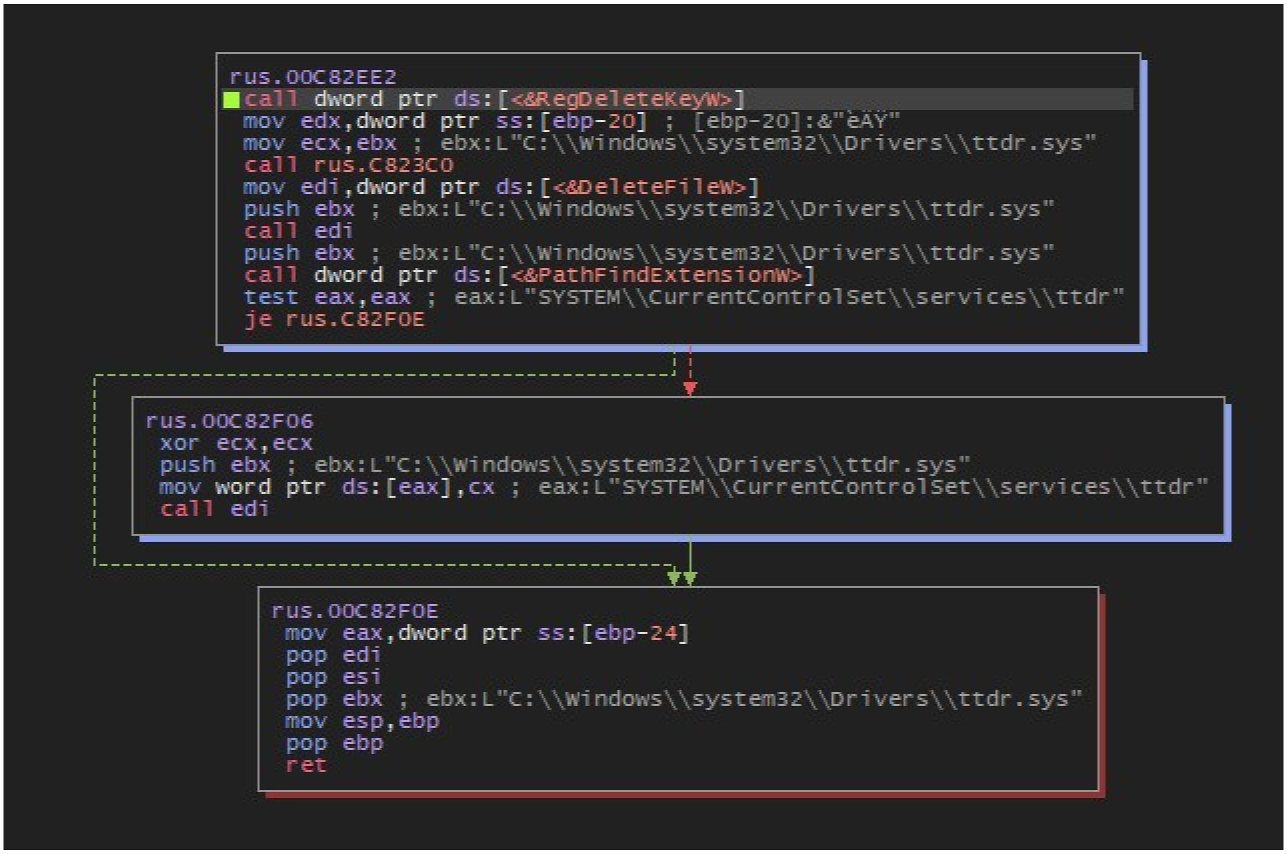
- If the service is present, then malware gets the service status from Service Control Manager using the *QueryServiceStatus* API.
- The service configurations are changed, if the service is inactive, via *ChangeServiceConfig* API. And the flag *SERVICE_DEMAND_START* (0x00000003) is passed as *dwStartType*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

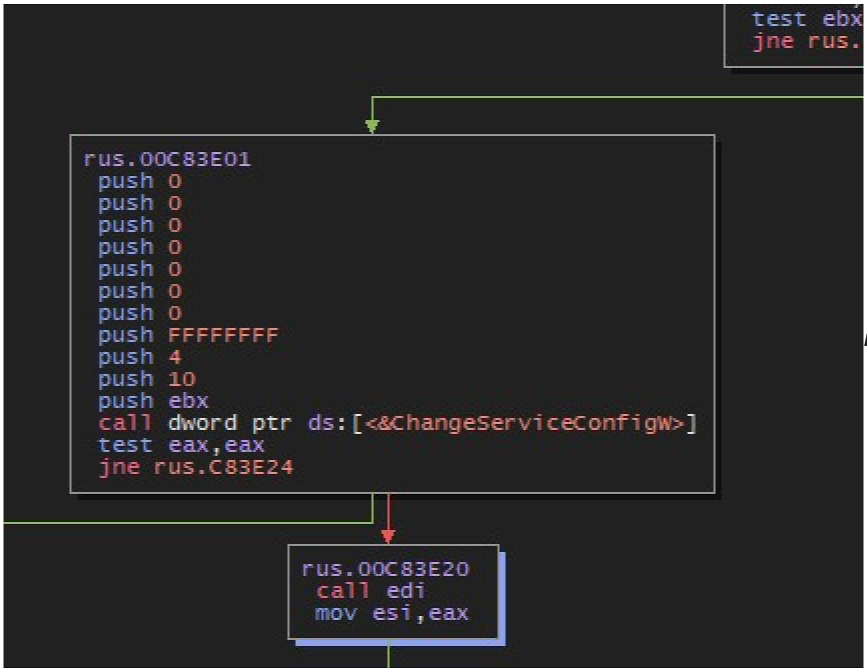


Deleting service entry from the registry and driver image

After the clean up process, the malware disables Volume Shadow Copy Service (VSS) on the system, via Service Control Manager. The VSS service is opened via the *OpenServiceW* API to change the configuration of the service later.

Disabling the Volume Shadow Copy Service

A new configuration update is made by passing the 0x00000004 flag (*SERVICE_DISABLED*) to *ChangeServiceConfigW,* thus disabling VSS by force.



Disabling VSS

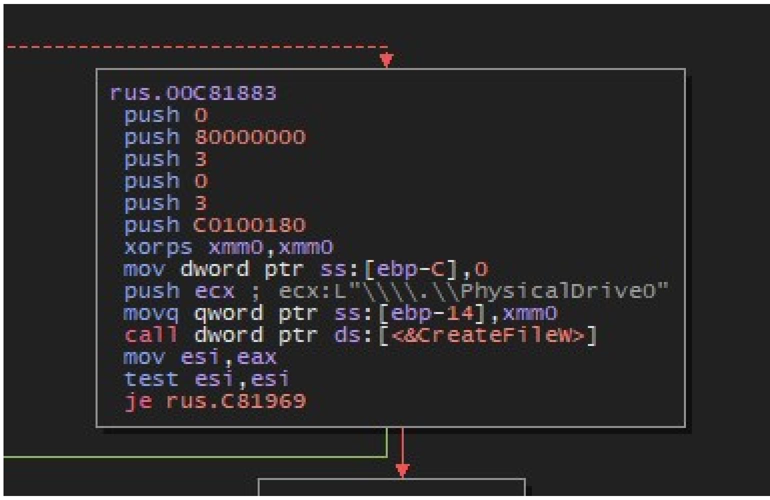
The malware makes sure the service has stopped, by passing 0x00000001 `(SERVICE_CONTROL_STOP)` as the *dwControl* parameter value.

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**EPMNTDRV),* 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	



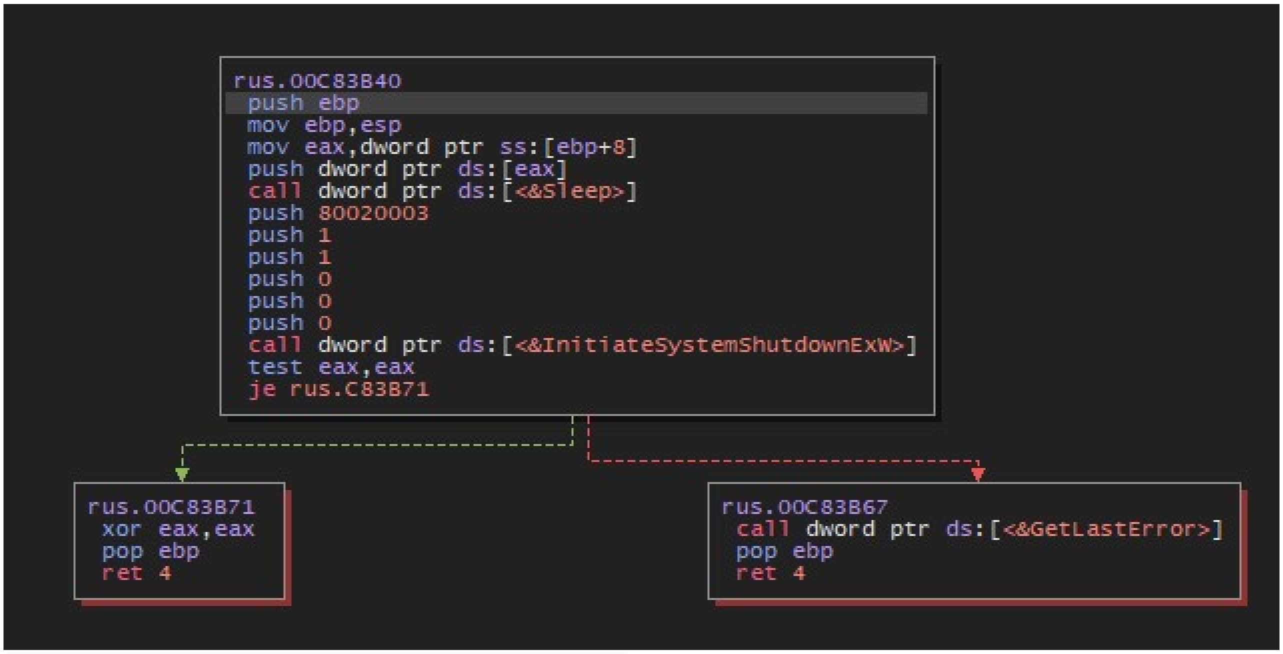
Accessing \\\\.\\PhysicalDrive0

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

NTFS corruption logic

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.



`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: 1bc44eef75779e3ca1eefb8ff5a64807dbc942b1e4a2672d77b9f6928d292591
- Dropped Driver: 6106653B08F4F72EEAA7F099E7C408A4
- 3F4A16B29F2F0532B7CE3E7656799125

Author Details

Anandeshwar Unnikrishnan

Threat Intelligence Researcher , CloudSEK
Anandeshwar is a Threat Intelligence Researcher at CloudSEK. He is a strong advocate of offensive cybersecurity. He is fuelled by his passion for cyber threats in a global context. He dedicates much of his time on Try Hack Me/ Hack The Box/ Offensive Security Playground. He believes that “a strong mind starts with a strong body.” When he is not gymming, he finds time to nurture his passion for teaching. He also likes to travel and experience new cultures.

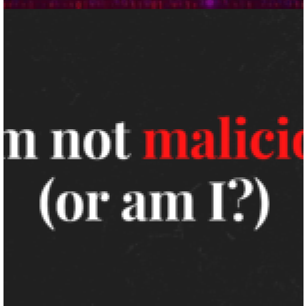
Deepanjli Paulraj
Lead Cyberintelligence Editor, CloudSEK
Total Posts: 3
Deepanjli is CloudSEK’s Lead Technical Content Writer and Editor. She is a pen wielding pedant with an insatiable appetite for books, Sudoku, and epistemology. She works on any and all content at CloudSEK, which includes blogs, reports, product documentation, and everything in between.

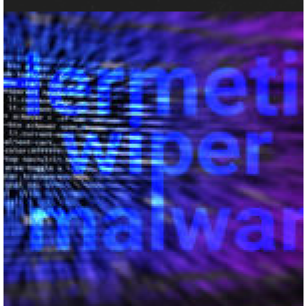
×

Anandeshwar Unnikrishnan
Threat Intelligence Researcher , CloudSEK
Anandeshwar is a Threat Intelligence Researcher at CloudSEK. He is a strong advocate of offensive cybersecurity. He is fuelled by his passion for cyber threats in a global context. He dedicates much of his time on Try Hack Me/ Hack The Box/ Offensive Security Playground. He believes that “a strong mind starts with a strong body.” When he is not gymming, he finds time to nurture his passion for teaching. He also likes to travel and experience new cultures.

Latest Posts

- 

- 

- 

-