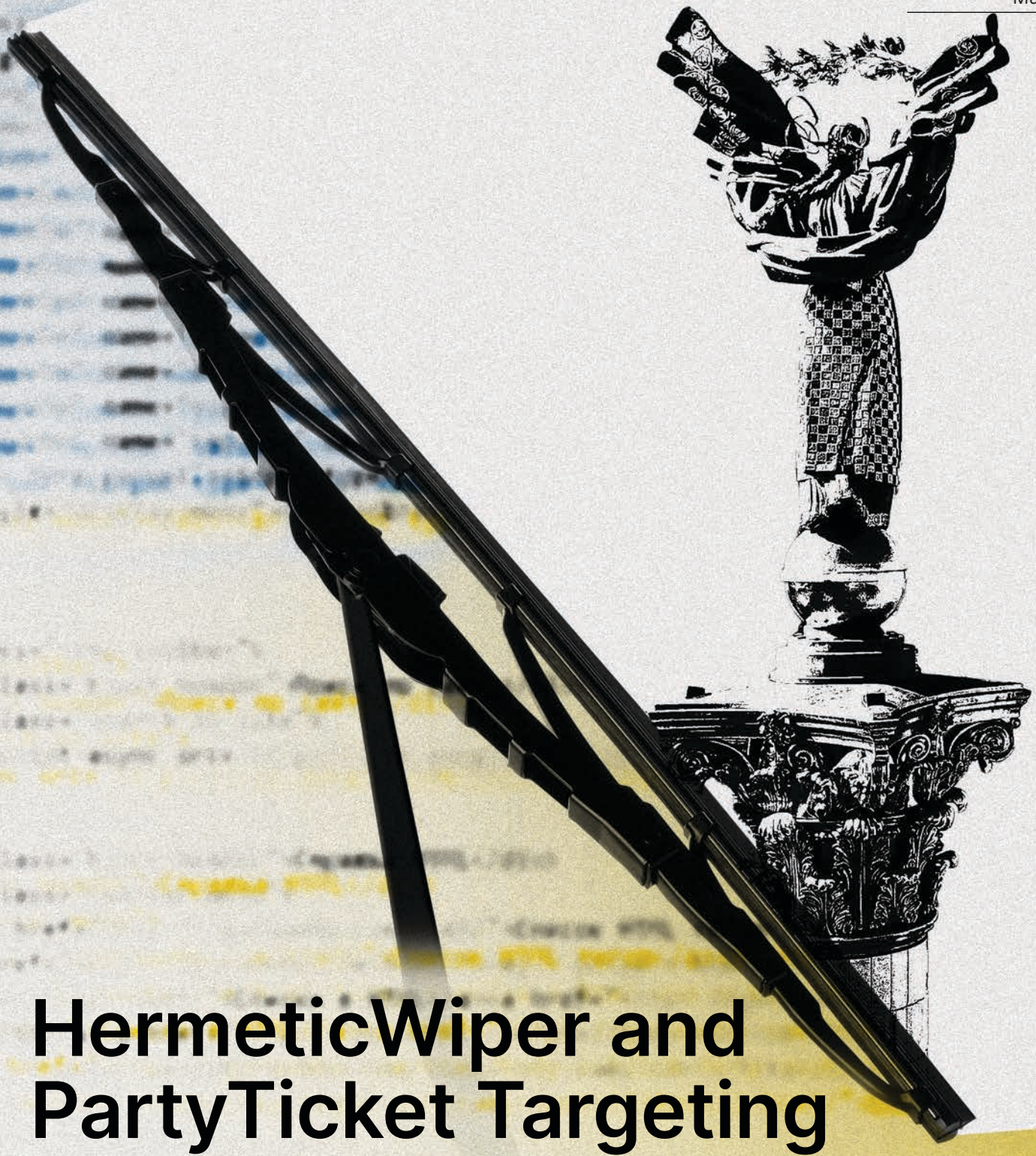
MALWARE/
TOOLS
PROFILE

Recorded Future®

By Insikt Group®

March 2, 2022



HermeticWiper and PartyTicket Targeting Computers in Ukraine

This report is a technical overview of the HermeticWiper and PartyTicket malware reported by ESET and Symantec on February 23, 2022. The malware was primarily delivered to Ukrainian organizations coincident with the Russian invasion of Ukraine. It is intended for those looking for a high-level overview of the malware's TTPs and mitigations.

Executive Summary

Insikt Group analyzed the HermeticWiper malware and the associated ransomware component named PartyTicket that were first publicly reported targeting Ukrainian organizations on February 23, 2022. We determined that both components serve the purpose of data destruction, with the "ransomware" component differing significantly in form and function from known criminal ransomware threats.

Key Judgments

- The use of a wiper malware with an associated destructive ransomware component is similar in method to WhisperGate, NotPetya, and other operations credited to Sandworm.
- There is insufficient evidence at this time to attribute HermeticWiper to the Russian state, but the timing of the mass deployment of HermeticWiper with kinetic attacks and other cyberattacks on Ukraine, and a methodology similar to past attacks by Russian government-associated actors, lends credence to such an attribution.
- The PartyTicket ransomware attacks are unlikely to be a true ransomware campaign conducted for financial gain. It is more likely that the ransomware component is a ruse and the real purpose of the attacks are disruption and data destruction.

Background

HermeticWiper, also known as FoxBlade, is a data wiper found targeting finance and government contractor organizations in Ukraine, Latvia, and Lithuania, according to Symantec.

ESET first reported that their telemetry indicated the malware was delivered to hundreds of systems in Ukraine, and the malware was executed on February 23, 2022, following DDoS attacks on Ukrainian websites earlier that day.

It was also reported by ESET that the malware was a signed executable, with a code-signing certificate issued to Hermetica Digital Ltd. Code-signing certificates allow malware to be more effectively deployed by bypassing detection capabilities, such as Microsoft Defender SmartScreen and built-in browser protections. The developer who operates Hermetica Digital Ltd. has publicly denied any involvement in the development of the malware, however.

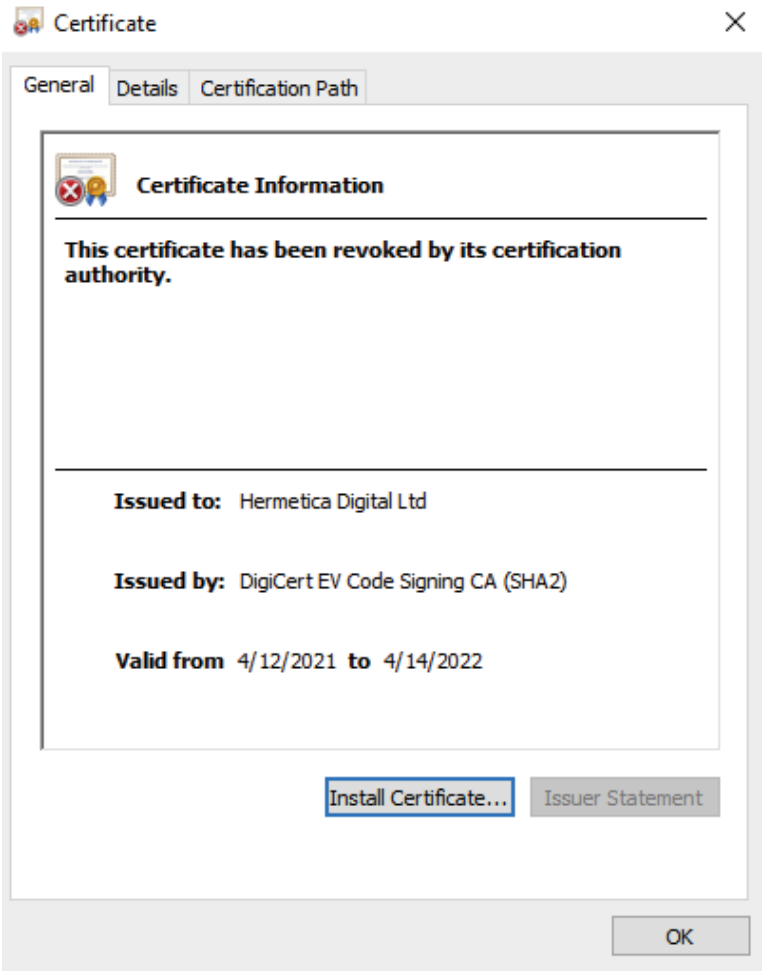


Figure 1: The now-revoked Hermetica Digital Ltd. code signing certificate used to sign HermeticWiper (Source: Recorded Future)

Microsoft reported that it identified the wiper attacks on February 24, 2022 and alerted the Ukrainian government.

Although no direct attribution of HermeticWiper has been made by security researchers at this time, the timing being coincident with other cyberattacks and physical attacks on Ukraine and a history of similar tactics by Russian state-associated actors in their use of data wipers in the past, including WhisperGate in January 2022 and NotPetya in 2017, suggest the involvement of a Russian state operation.

Technical Analysis

Multiple infection vectors for the HermeticWiper malware have been reported by Symantec.

The delivery of the malware to a Ukrainian organization followed a Server Message Block (SMB)-based attack on a Microsoft Exchange server on December 23, 2021. The adversary initially stole credentials, and a web shell was installed on January 16, 2022. HermeticWiper was finally deployed on February 23, 2022.

A Lithuanian organization that received HermeticWiper was initially compromised in November 2021. The delivery mechanism was suspected by Symantec as being an Apache Tomcat exploit that executed a malicious PowerShell command. This attack similarly included a credential harvesting component, followed quickly by the delivery and execution of the wiper malware as a scheduled task.

In several of the attacks, a ransomware executable was delivered alongside HermeticWiper. The victims received a ransomware notification providing 2 email addresses: vote2024forjb@protonmail[.]com and stephanie.jones2024@protonmail[.]com. Symantec considers it likely that the ransomware component was used to distract the victims. WhisperGate attacks used a similar methodology, wherein the attack was disguised as ransomware. Both WhisperGate and HermeticWiper used separate components to prevent a victim's system from booting and file corruption; however, the component that played the role of ransomware changed between the 2 attacks. With WhisperGate, the wiper itself masqueraded as ransomware; however, with the HermeticWiper attacks, it was the file corrupter instead.

ESET reported that in one instance they observed, the malware was dropped via default Group Policy Objects (GPO), indicating that the adversaries almost certainly had control of an Active Directory server on the network.

HermeticWiper

HermeticWiper's primary purpose is to corrupt the NTFS and/or FAT file systems of a victim's machine to prevent it from booting correctly. It was written in Visual Studio 2008 and 2015 in a combination of C and assembly and uses an included kernel driver to implement much of its disk access functionality. The use of a kernel driver instead of conventional Windows API calls is thought to evade detections that may catch the higher-level API calls being made. The compiler timestamps for 2 samples show that they were compiled on December 28, 2021, and 1 other sample shows February 23, 2022. Although timestamps can be forged, the timestamp from December 28, 2021, could be used to determine how far in advance this operation was planned. Each sample is signed using what was, at the time, a valid certificate issued to Hermetica Digital Ltd. Since the malware's discovery, the certificate has since been revoked by the Certificate Authority, as shown in Figure 1 above.

Upon execution, the wiper adjusts its process token privileges to acquire SeBackupPrivilege and SeShutdownPrivilege in order to obtain read privileges to any files and eventually shut down the system before terminating. Next, it determines the Windows version and bitness (x86 or x86_64) of the victim's machine in order to determine which kernel driver, located in the PE's resource section, to later load. The kernel drivers are legitimate, benign software used by EaseUS's Partition Master and are signed with a certificate issued to EaseUS's parent company, CHENGDU YIWO Tech Development Co., Ltd., shown in Figure 2 below. Although the certificate has expired, newer versions of Windows 10 allow exceptions for kernel drivers with certificates issued before July 29, 2015, to be loaded.

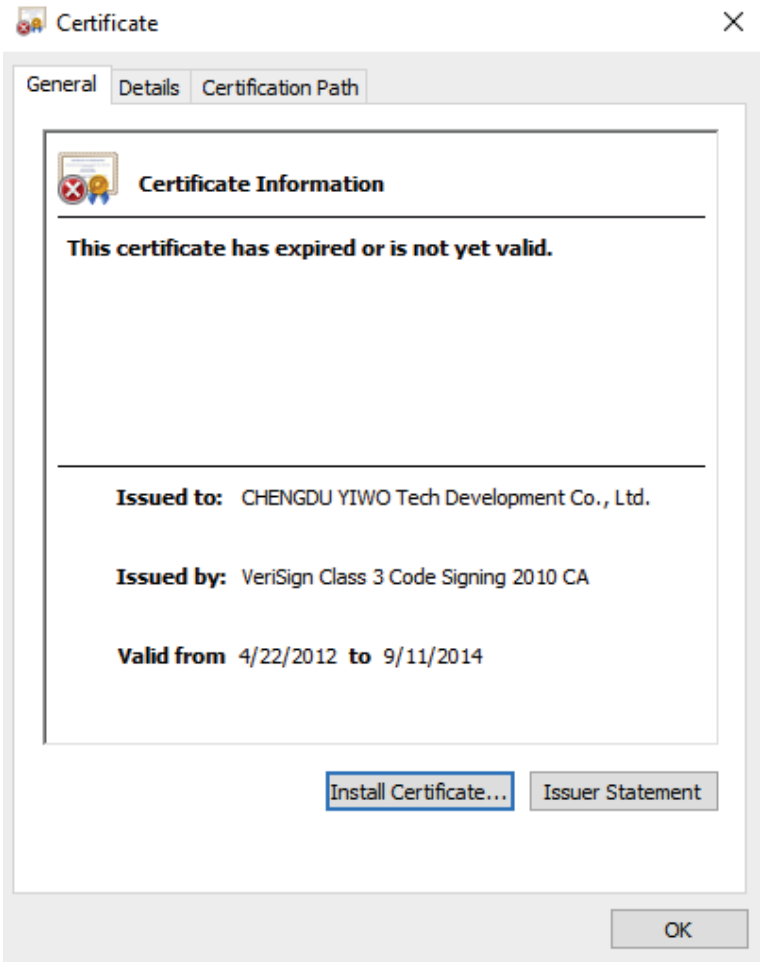


Figure 2: Certificate used to sign EaseUS Partition Master drivers (Source: Recorded Future)

The kernel drivers are stored as RCDATA in the PE file's resource section, as shown in Figure 3 below. Each driver is compressed using Microsoft's SZDD file format, based on the Lempel–Ziv–Storer–Szymanski (LZSS) algorithm.

After locating the correct driver, the wiper then proceeds to disable WoW64 File System Redirection if the victim is running a 64-bit OS. This prevents 64-bit systems from loading 32-bit kernel drivers from the %windir%\SysWOW64\drivers directory and instead forces them to use %windir%\system32\drivers, where the malware will eventually place the kernel driver. Next, Crash Dumps are disabled by modifying the registry value "CrashDumpEnabled" to 0 for the key HKLM\SYSTEM\CurrentControlSet\Control\CrashControl. This is likely done to avoid writing a crash dump to disk when the program terminates.

The compressed driver resource is then written to the %windir%\system32\drivers directory with a name consisting of 2 pseudorandom lowercase characters followed by "dr". Then the driver is decompressed, and a service with the same name is temporarily created to load the driver.

To create the service, the process's token privileges are adjusted again to add SeLoadDriverPrivilege. A service is then created, configured, and started. Once the driver is successfully loaded, the created service's registry entry is removed from HKLM\SYSTEM\CurrentControlSet\services\. Next, the Volume Shadow Service (VSS) is stopped and disabled, as shown in Figure 4, to make recovery more difficult.

The wiper then begins to iterate through all physical drives on the system one at a time by attempting to access \\.\PhysicalDrive<1-100>. For each drive, junk data is written to seemingly random locations of the disk in order to corrupt it. Additionally, the partitions on each physical disk are enumerated and any identified as FAT or NTFS file systems are corrupted by writing random data to the file system header. Although public reporting has stated that the MBR is "wiped", which typically means the MBR is overwritten, in our analysis we have concluded that only the file system is corrupted along with random locations on the disk. The end result similarly results in a loss of the stored data and inability of the victim machine to boot. Appendix A provides the output from a tool, API Monitor, that captured the SetFilePointerEX and WriteFile API calls used to corrupt the hard drives on the victim's machine. There were no writes to the "0" index, where the MBR would reside, however there are writes to the index, 1048576, which is the location of the NTFS file system header. The additional writes are to seemingly random locations on disk.

The corruption of the file systems goes beyond a simple MBR overwrite and is more effective because it impacts a victim's ability to boot regardless of the disk partitioning scheme (i.e., MBR, GPT). This technique is more robust than the MBR overwrite used in the WhisperGate attacks, where we showed that GPT-style disks could recover from the MBR overwrite.

After corrupting the file system the wiper disables the ShowCompColor and ShowInfoTip values in the Software\Microsoft\CurrentVersion\Explorer\Advanced registry key in order to prevent encrypted NTFS files from showing in color and showing pop-up descriptions for folders, respectively. Then it proceeds to corrupt logs and data on NTFS file systems.

Finally, the wiper attempts to shut the system down with a call to InitiateSystemShutdownExW. Once the victim machine is rebooted, the user is presented with an error message indicating that their system cannot boot. In the case of MBR-style disks, the victim is presented with a message similar to the one shown in Figure 5; or in the case of GPT-style disks, the one shown in Figure 6. In both cases, although the MBR is still intact, the system is unable to boot due to the corrupted file system partition containing the Operating System.

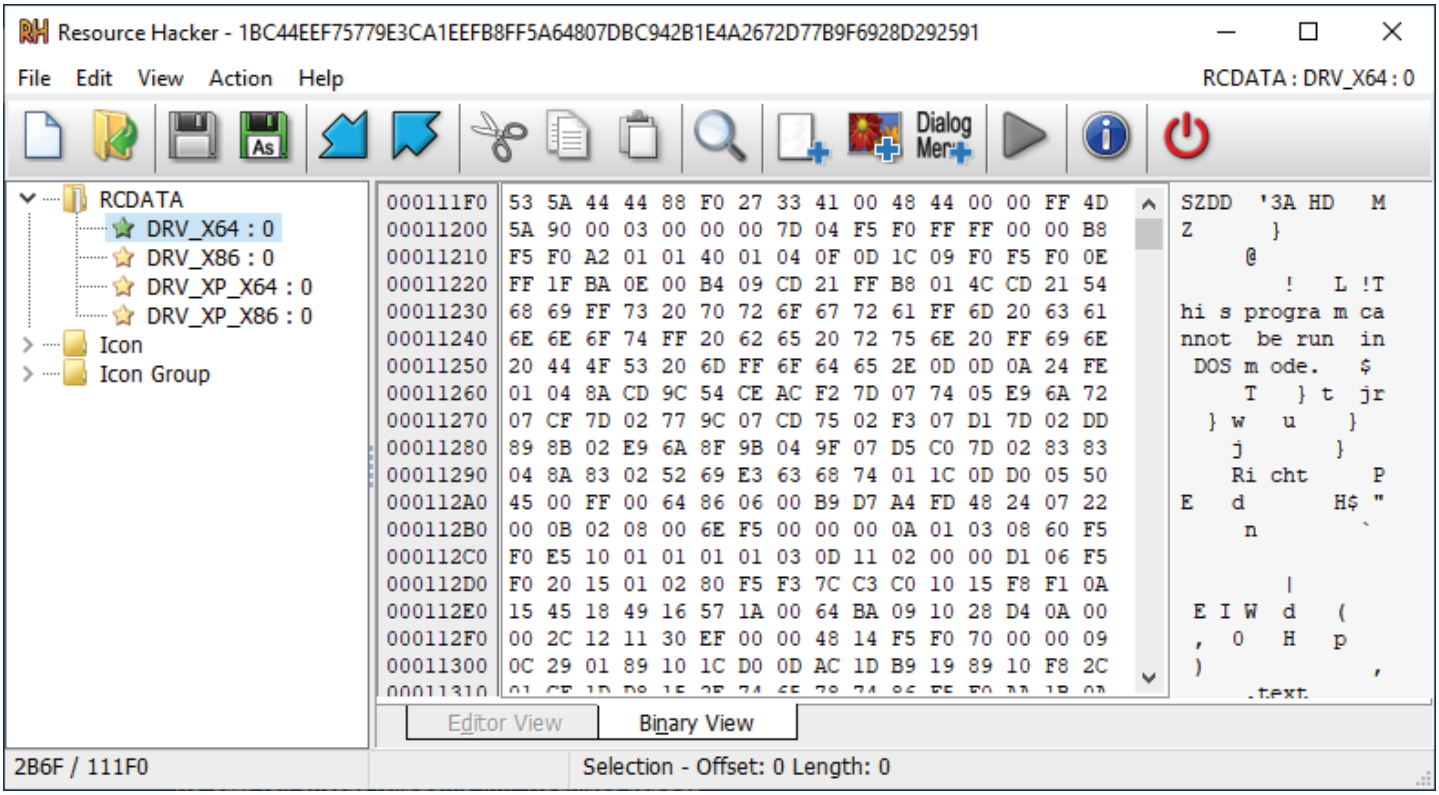


Figure 3: SZDD compressed resources stored in the resource section of the wiper (Source: Recorded Future)

```
if (driverLoaded != 0) {
  lastError = 0;
  hSCManager = OpenSCManagerW((LPCWSTR)0x0,L"ServicesActive",SC_MANAGER_ALL_ACCESS);
  if (hSCManager == (SC_HANDLE)0x0) {
    lastError = (*getLastErrorPtr)();
  }
  else {
    vssService = OpenServiceW(hSCManager,L"vss",0x22);
    if (vssService == (SC_HANDLE)0x0) {
      lastError = (*getLastErrorPtr)();
      closeServiceHandlePtr = CloseServiceHandle_exref;
    }
    else {
      uStackY1384 = 0x153e1c;
      retVal = ChangeServiceConfigW
                         (vssService,SERVICE_WIN32_OWN_PROCESS,SERVICE_DISABLED,0xffffffff,
                          (LPCWSTR)0x0,(LPCWSTR)0x0,(LPDWORD)0x0,(LPCWSTR)0x0,(LPCWSTR)0x0,
                          (LPCWSTR)0x0,(LPCWSTR)0x0);
      if (retVal == 0) {
        lastError = (*getLastErrorPtr)();
      }
      ControlService(vssService,SERVICE_CONTROL_STOP,(LPSERVICE_STATUS)0x0);
      closeServiceHandlePtr = CloseServiceHandle_exref;
      CloseServiceHandle(vssService);
```

Figure 4: Disabling and stopping the VSS service (Source: Recorded Future)

Figure 5: Boot screen after infection on a MBR disk (Source: Recorded Future)

Figure 6: Boot screen after infection on a GPT disk (Source: Recorded Future)

PartyTicket

Insikt Group analyzed the ransomware associated with the HermeticWiper malware, dubbed PartyTicket. The ransomware contained several path strings and function names that allude to the White House, Joe Biden, and elections, among other topics, seen below in Figures 7 and 8.

```
"_/C_/projects/403forBiden/wHiteHousE.init"
"_/C_/projects/403forBiden/wHiteHousE.FileName"
"_/C_/projects/403forBiden/wHiteHousE.statictmp_0"
"_/C_/projects/403forBiden/wHiteHousE.baggageGatherings"
"_/C_/projects/403forBiden/wHiteHousE.lookUp"
"_/C_/projects/403forBiden/wHiteHousE.primaryElectionProcess"
"_/C_/projects/403forBiden/wHiteHousE.GoodOffice1"
```

Figure 7: Paths contained in the ransomware (Source: Recorded Future)

main.dtFie	00500f20
main.pr1me	00500f70
main.getBoo	00501120
main.n1hk9	00501210
main.bUnderPk	005016d0
main.selfElect	005019f0
main.randomiseDuration	00501d00
main.highWay60	00501d50
main.voteFor403	00501ff0
main.init	00502440

Figure 8: Ransomware function names (Source: Recorded Future)

Similarly, the ransom note dropped by the malware contained email addresses on similar topics, and the encrypted files were renamed with the suffix "[vote2024forjb@protonmail[.]com].encryptedJB", as shown in Figure 10.

Figure 9: Example of encrypted file with extension (Source: Recorded Future)

Figure 10 below shows the ransomware note dropped by PartyTicket. While Insikt Group cannot currently attribute the ransomware to any specific group, the note differs substantially from that of other ransomware groups we have seen.

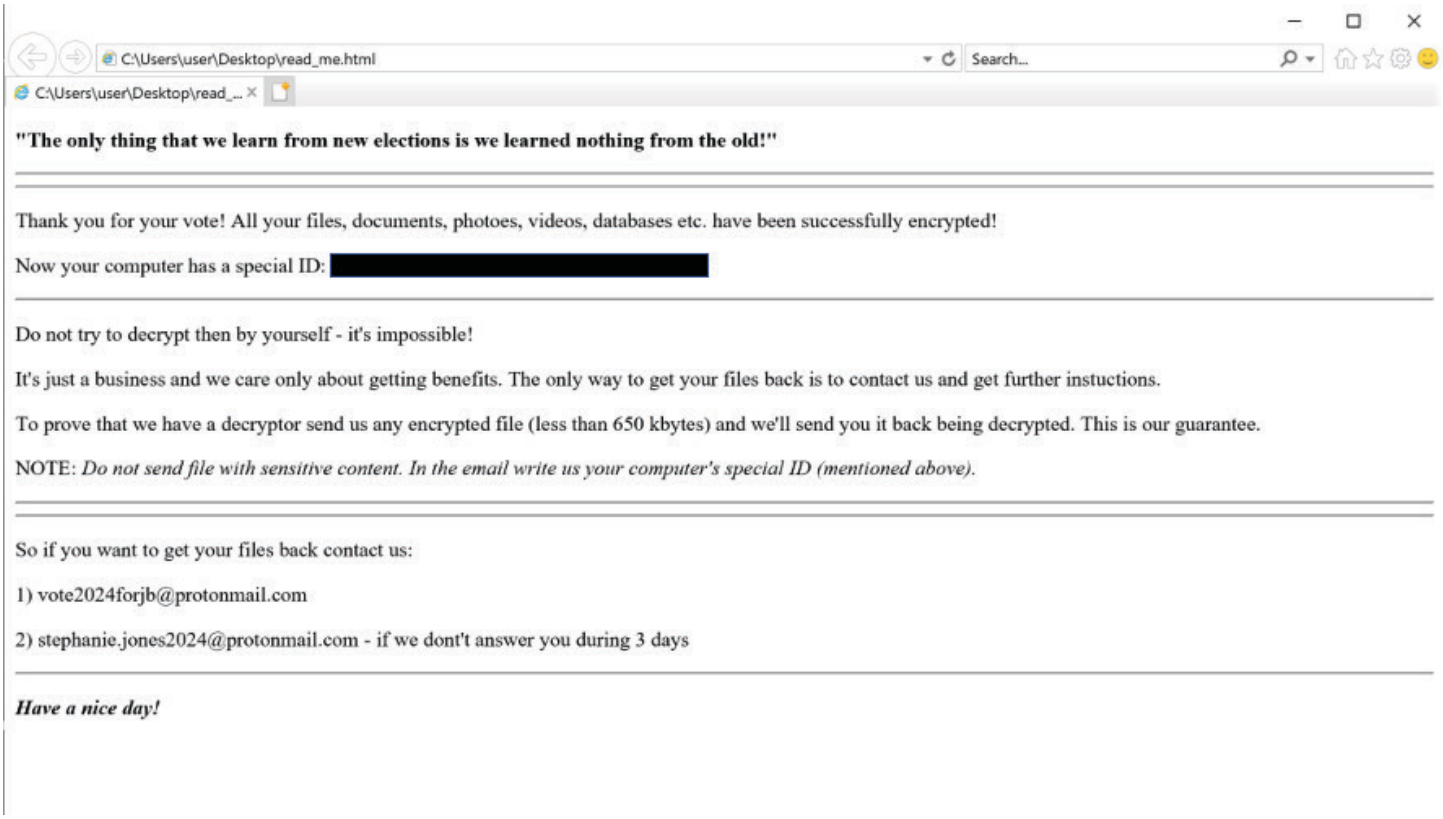


Figure 10: Ransom note (Source: Recorded Future)

There is no "branding" identifying a particular ransomware group responsible for the attack, and there are several misspellings and grammatical errors throughout the note. As a result, the ransomware component of the HermeticWiper malware is unlikely to have been developed and distributed by a criminal ransomware group. Further, the malware contains a list of files, shown in Figure 12 below, that it seeks to encrypt. Unlike all other recent, criminal-operated ransomware variants, this list includes files that are key to the ability of the victim system to operate, including .dll and .exe files. This further suggests that this is not legitimate ransomware but rather a destructive piece of malware.

```
-inf .acl, .avi .bat .bmp .cab .cfg .chm .cmd .com .crt .css
.dat .dip .dll .doc .dot .exe .gif .htm .ico .iso .jpg .mp3 .msi
.odt .one .ova .pdf .png .ppt .pub .rar .rtf .sfx .sql .txt .url .vdi
.vsd .wma .wmv .wtv .xls .xml .xps .zip
```

Figure 11: File extensions the "ransomware" seeks to encrypt (Source: Recorded Future)

Mitigations

The compromised systems leading to the delivery of the wiper have involved exploitation of vulnerable systems: a Microsoft Exchange server, and an Apache Tomcat server. Defenders concerned specifically about HermeticWiper should ensure that any such servers on their networks are fully updated and patched. Similarly, enterprises should prioritize detection of web shells and exploitation on their perimeters. Detection of a wiper malware at the point of execution is often too late in the kill chain to ensure continued organizational operations. Focusing on the initial stages is important to avoid such malware being executed.

Endeavoring to prevent, detect and block early-stage activity observed in the delivery of HermeticWiper, such as malicious PowerShell usage and SMB exploitation, is thus recommended. The adversary, nimble enough to exploit more than one type of system to deliver HermeticWiper, is likely capable of delivering malware to other vulnerable systems as well, and consistent patching and updating of all external-facing systems is therefore critical.

On February 26, 2022, CISA issued an alert concerning the use of destructive malware, specifically HermeticWiper and WhisperGate, against Ukrainian organizations. General best practices and mitigations for wiper malware are provided in the alert. By keeping updated on the current situation in Ukraine, in particular in the cyber realm, an organization can better prioritize patching and other mitigations based on what is currently known of potential threats.

Insikt Group has provided 2 YARA rules to detect HermeticWiper and PartyTicket in Appendix B.

Outlook

This is the second destructive malware that has emerged over the past month, coinciding with the timing of attacks on Ukraine, and exhibiting a methodology similar to past attacks by Russian government-associated actors. We expect further cyberattacks or malicious tools to emerge and be used to destroy data and cause other disruptions. While there is not enough evidence to tie either of these wipers to a specific threat actor or group, HermeticWiper's similarities to previous Russian state-linked malware variants, such as NotPetya, could suggest some relationship.

Appendix A: SetFilePointerEx and WriteFile Windows API Calls

The table below shows the output from the tool API Monitor and was specifically capturing the API calls SetFilePointerEx, WriteFile, and wnsprintfw. To get the drive index location that SetFilePointerEx is pointing to, you must combine the "HighPart" and "LowPart" values to get the full index. For example, the full index for the call, "SetFilePointerEx (0×0000026c, { u = { LowPart = 2539999232, HighPart = 17 }, QuadPart = 75554443264 }, NULL, FILE_BEGIN)" would be 172539999232.

Module	API
1bc44.exe	wnsprintfW ("", 260, "\\.\EPMNTDRV\%u", ...)
1bc44.exe	SetFilePointerEx (0×0000026c, { u = { LowPart = 2539999232, HighPart = 17 }, QuadPart = 75554443264 }, NULL, FILE_BEGIN)
1bc44.exe	WriteFile (0×0000026c, 0×01587470, 4096, 0×044dfa68, NULL)
1bc44.exe	SetFilePointerEx (0×0000026c, { u = { LowPart = 2540003328, HighPart = 17 }, QuadPart = 75554447360 }, NULL, FILE_BEGIN)
1bc44.exe	WriteFile (0×0000026c, 0×01587470, 4096, 0×044dfa68, NULL)
1bc44.exe	SetFilePointerEx (0×0000026c, { u = { LowPart = 2540007424, HighPart = 17 }, QuadPart = 75554451456 }, NULL, FILE_BEGIN)
1bc44.exe	WriteFile (0×0000026c, 0×01587470, 4096, 0×044dfa68, NULL)
1bc44.exe	SetFilePointerEx (0×0000026c, { u = { LowPart = 2540011520, HighPart = 17 }, QuadPart = 75554455552 }, NULL, FILE_BEGIN)
1bc44.exe	WriteFile (0×0000026c, 0×01587470, 4096, 0×044dfa68, NULL)
1bc44.exe	SetFilePointerEx (0×0000026c, { u = { LowPart = 2540015616, HighPart = 17 }, QuadPart = 75554459648 }, NULL, FILE_BEGIN)
1bc44.exe	WriteFile (0×0000026c, 0×01587470, 4096, 0×044dfa68, NULL)
1bc44.exe	SetFilePointerEx (0×0000026c, { u = { LowPart = 2540019712, HighPart = 17 }, QuadPart = 75554463744 }, NULL, FILE_BEGIN)
1bc44.exe	WriteFile (0×0000026c, 0×01587470, 4096, 0×044dfa68, NULL)
1bc44.exe	SetFilePointerEx (0×0000026c, { u = { LowPart = 2540023808, HighPart = 17 }, QuadPart = 75554467840 }, NULL, FILE_BEGIN)
1bc44.exe	WriteFile (0×0000026c, 0×01587470, 4096, 0×044dfa68, NULL)
1bc44.exe	SetFilePointerEx (0×0000026c, { u = { LowPart = 2540027904, HighPart = 17 }, QuadPart = 75554471936 }, NULL, FILE_BEGIN)
1bc44.exe	WriteFile (0×0000026c, 0×01587470, 4096, 0×044dfa68, NULL)
1bc44.exe	SetFilePointerEx (0×0000026c, { u = { LowPart = 2540032000, HighPart = 17 }, QuadPart = 75554476032 }, NULL, FILE_BEGIN)
1bc44.exe	WriteFile (0×0000026c, 0×01587470, 4096, 0×044dfa68, NULL)
1bc44.exe	SetFilePointerEx (0×0000026c, { u = { LowPart = 2540036096, HighPart = 17 }, QuadPart = 75554480128 }, NULL, FILE_BEGIN)
1bc44.exe	WriteFile (0×0000026c, 0×01587470, 4096, 0×044dfa68, NULL)
1bc44.exe	SetFilePointerEx (0×0000026c, { u = { LowPart = 2540040192, HighPart = 17 }, QuadPart = 75554484224 }, NULL, FILE_BEGIN)
1bc44.exe	WriteFile (0×0000026c, 0×01587470, 4096, 0×044dfa68, NULL)
1bc44.exe	SetFilePointerEx (0×0000026c, { u = { LowPart = 2540044288, HighPart = 17 }, QuadPart = 75554488320 }, NULL, FILE_BEGIN)

Module	API
1bc44.exe	WriteFile (0×0000026c, 0×01587470, 4096, 0×044dfa68, NULL)
1bc44.exe	SetFilePointerEx (0×0000026c, { u = { LowPart = 2540048384, HighPart = 17 }, QuadPart = 75554492416 }, NULL, FILE_BEGIN)
1bc44.exe	WriteFile (0×0000026c, 0×01587470, 4096, 0×044dfa68, NULL)
1bc44.exe	SetFilePointerEx (0×0000026c, { u = { LowPart = 2540052480, HighPart = 17 }, QuadPart = 75554496512 }, NULL, FILE_BEGIN)
1bc44.exe	WriteFile (0×0000026c, 0×01587470, 4096, 0×044dfa68, NULL)
1bc44.exe	SetFilePointerEx (0×0000026c, { u = { LowPart = 2540056576, HighPart = 17 }, QuadPart = 75554500608 }, NULL, FILE_BEGIN)
1bc44.exe	WriteFile (0×0000026c, 0×01587470, 4096, 0×044dfa68, NULL)
1bc44.exe	SetFilePointerEx (0×0000026c, { u = { LowPart = 2540060672, HighPart = 17 }, QuadPart = 75554504704 }, NULL, FILE_BEGIN)
1bc44.exe	WriteFile (0×0000026c, 0×01587470, 4096, 0×044dfa68, NULL)
1bc44.exe	SetFilePointerEx (0×0000026c, { u = { LowPart = 2540064768, HighPart = 17 }, QuadPart = 75554508800 }, NULL, FILE_BEGIN)
1bc44.exe	WriteFile (0×0000026c, 0×01587470, 4096, 0×044dfa68, NULL)
1bc44.exe	SetFilePointerEx (0×0000026c, { u = { LowPart = 2540068864, HighPart = 17 }, QuadPart = 75554512896 }, NULL, FILE_BEGIN)
1bc44.exe	WriteFile (0×0000026c, 0×01587470, 4096, 0×044dfa68, NULL)
1bc44.exe	SetFilePointerEx (0×0000026c, { u = { LowPart = 2540072960, HighPart = 17 }, QuadPart = 75554516992 }, NULL, FILE_BEGIN)
1bc44.exe	WriteFile (0×0000026c, 0×01587470, 4096, 0×044dfa68, NULL)
1bc44.exe	SetFilePointerEx (0×0000026c, { u = { LowPart = 2540077056, HighPart = 17 }, QuadPart = 75554521088 }, NULL, FILE_BEGIN)
1bc44.exe	WriteFile (0×0000026c, 0×01587470, 4096, 0×044dfa68, NULL)
1bc44.exe	SetFilePointerEx (0×0000026c, { u = { LowPart = 2540081152, HighPart = 17 }, QuadPart = 75554525184 }, NULL, FILE_BEGIN)
1bc44.exe	WriteFile (0×0000026c, 0×01587470, 4096, 0×044dfa68, NULL)
1bc44.exe	SetFilePointerEx (0×0000026c, { u = { LowPart = 2540085248, HighPart = 17 }, QuadPart = 75554529280 }, NULL, FILE_BEGIN)
1bc44.exe	WriteFile (0×0000026c, 0×01587470, 4096, 0×044dfa68, NULL)
1bc44.exe	SetFilePointerEx (0×0000026c, { u = { LowPart = 2540089344, HighPart = 17 }, QuadPart = 75554533376 }, NULL, FILE_BEGIN)
1bc44.exe	WriteFile (0×0000026c, 0×01587470, 4096, 0×044dfa68, NULL)
1bc44.exe	SetFilePointerEx (0×0000026c, { u = { LowPart = 2540093440, HighPart = 17 }, QuadPart = 75554537472 }, NULL, FILE_BEGIN)
1bc44.exe	WriteFile (0×0000026c, 0×01587470, 4096, 0×044dfa68, NULL)
1bc44.exe	SetFilePointerEx (0×0000026c, { u = { LowPart = 2540097536, HighPart = 17 }, QuadPart = 75554541568 }, NULL, FILE_BEGIN)
1bc44.exe	WriteFile (0×0000026c, 0×01587470, 4096, 0×044dfa68, NULL)

Module	API
1bc44.exe	SetFilePointerEx (0×0000026c, { u = { LowPart = 2540101632, HighPart = 17 }, QuadPart = 75554545664 }, NULL, FILE_BEGIN)
1bc44.exe	WriteFile (0×0000026c, 0×01587470, 4096, 0×044dfa68, NULL)
1bc44.exe	SetFilePointerEx (0×0000026c, { u = { LowPart = 2540105728, HighPart = 17 }, QuadPart = 75554549760 }, NULL, FILE_BEGIN)
1bc44.exe	WriteFile (0×0000026c, 0×01587470, 4096, 0×044dfa68, NULL)
1bc44.exe	SetFilePointerEx (0×0000026c, { u = { LowPart = 2540109824, HighPart = 17 }, QuadPart = 75554553856 }, NULL, FILE_BEGIN)
1bc44.exe	WriteFile (0×0000026c, 0×01587470, 4096, 0×044dfa68, NULL)
1bc44.exe	SetFilePointerEx (0×0000026c, { u = { LowPart = 2540113920, HighPart = 17 }, QuadPart = 75554557952 }, NULL, FILE_BEGIN)
1bc44.exe	WriteFile (0×0000026c, 0×01587470, 4096, 0×044dfa68, NULL)
1bc44.exe	SetFilePointerEx (0×0000026c, { u = { LowPart = 393588736, HighPart = 0 }, QuadPart = 393588736 }, NULL, FILE_BEGIN)
1bc44.exe	WriteFile (0×0000026c, 0×01587470, 4096, 0×044dfa68, NULL)
1bc44.exe	SetFilePointerEx (0×0000026c, { u = { LowPart = 393592832, HighPart = 0 }, QuadPart = 393592832 }, NULL, FILE_BEGIN)
1bc44.exe	WriteFile (0×0000026c, 0×01587470, 4096, 0×044dfa68, NULL)
1bc44.exe	SetFilePointerEx (0×0000026c, { u = { LowPart = 393596928, HighPart = 0 }, QuadPart = 393596928 }, NULL, FILE_BEGIN)
1bc44.exe	WriteFile (0×0000026c, 0×01587470, 4096, 0×044dfa68, NULL)
1bc44.exe	SetFilePointerEx (0×0000026c, { u = { LowPart = 393601024, HighPart = 0 }, QuadPart = 393601024 }, NULL, FILE_BEGIN)
1bc44.exe	WriteFile (0×0000026c, 0×01587470, 4096, 0×044dfa68, NULL)
1bc44.exe	SetFilePointerEx (0×0000026c, { u = { LowPart = 393605120, HighPart = 0 }, QuadPart = 393605120 }, NULL, FILE_BEGIN)
1bc44.exe	WriteFile (0×0000026c, 0×01587470, 4096, 0×044dfa68, NULL)
1bc44.exe	SetFilePointerEx (0×0000026c, { u = { LowPart = 2068054016, HighPart = 0 }, QuadPart = 2068054016 }, NULL, FILE_BEGIN)
1bc44.exe	WriteFile (0×0000026c, 0×01587470, 4096, 0×044dfa68, NULL)
1bc44.exe	SetFilePointerEx (0×0000026c, { u = { LowPart = 2068058112, HighPart = 0 }, QuadPart = 2068058112 }, NULL, FILE_BEGIN)
1bc44.exe	WriteFile (0×0000026c, 0×01587470, 4096, 0×044dfa68, NULL)
1bc44.exe	SetFilePointerEx (0×0000026c, { u = { LowPart = 2068062208, HighPart = 0 }, QuadPart = 2068062208 }, NULL, FILE_BEGIN)
1bc44.exe	WriteFile (0×0000026c, 0×01587470, 4096, 0×044dfa68, NULL)
1bc44.exe	wnsprintfW ("", 260, "\\.\PhysicalDrive%u", ...)
1bc44.exe	wnsprintfW ("", 260, "\\.\EPMNTDRV\%u", ...)
1bc44.exe	SetFilePointerEx (0×00000204, { u = { LowPart = 1048576, HighPart = 0 }, QuadPart = 1048576 }, NULL, FILE_BEGIN)

Module	API
1bc44.exe	WriteFile (0×00000204, 0×0158ef48, 4096, 0×06b2f630, NULL)
1bc44.exe	wnsprintfW ("", 260, "\\.\EPMNTDRV\%u", ...)
1bc44.exe	SetFilePointerEx (0×00000214, { u = { LowPart = 3566206976, HighPart = 0 }, QuadPart = 3566206976 }, NULL, FILE_BEGIN)
1bc44.exe	wnsprintfW ("\\.\PhysicalDrive0", 260, "\\.\PhysicalDrive%u", ...)
1bc44.exe	SetFilePointerEx (0×00000248, { u = { LowPart = 2429767680, HighPart = 2 }, QuadPart = 11019702272 }, NULL, FILE_BEGIN)
1bc44.exe	WriteFile (0×00000248, 0×0158e538, 2048, 0×06c6f820, NULL)
1bc44.exe	SetFilePointerEx (0×00000248, { u = { LowPart = 2429769728, HighPart = 2 }, QuadPart = 11019704320 }, NULL, FILE_BEGIN)
1bc44.exe	WriteFile (0×00000248, 0×0158e538, 2048, 0×06c6f820, NULL)
1bc44.exe	SetFilePointerEx (0×00000248, { u = { LowPart = 2429771776, HighPart = 2 }, QuadPart = 11019706368 }, NULL, FILE_BEGIN)
1bc44.exe	WriteFile (0×00000248, 0×0158e538, 2048, 0×06c6f820, NULL)
1bc44.exe	SetFilePointerEx (0×00000248, { u = { LowPart = 2429773824, HighPart = 2 }, QuadPart = 11019708416 }, NULL, FILE_BEGIN)
1bc44.exe	WriteFile (0×00000248, 0×0158e538, 2048, 0×06c6f820, NULL)
1bc44.exe	SetFilePointerEx (0×00000248, { u = { LowPart = 2429775872, HighPart = 2 }, QuadPart = 11019710464 }, NULL, FILE_BEGIN)
1bc44.exe	WriteFile (0×00000248, 0×0158e538, 2048, 0×06c6f820, NULL)
1bc44.exe	SetFilePointerEx (0×00000248, { u = { LowPart = 2429777920, HighPart = 2 }, QuadPart = 11019712512 }, NULL, FILE_BEGIN)
1bc44.exe	WriteFile (0×00000248, 0×0158e538, 2048, 0×06c6f820, NULL)
1bc44.exe	SetFilePointerEx (0×00000248, { u = { LowPart = 2429779968, HighPart = 2 }, QuadPart = 11019714560 }, NULL, FILE_BEGIN)
1bc44.exe	WriteFile (0×00000248, 0×0158e538, 2048, 0×06c6f820, NULL)
1bc44.exe	SetFilePointerEx (0×00000248, { u = { LowPart = 2429782016, HighPart = 2 }, QuadPart = 11019716608 }, NULL, FILE_BEGIN)
1bc44.exe	WriteFile (0×00000248, 0×0158e538, 2048, 0×06c6f820, NULL)
1bc44.exe	SetFilePointerEx (0×00000248, { u = { LowPart = 2429784064, HighPart = 2 }, QuadPart = 11019718656 }, NULL, FILE_BEGIN)
1bc44.exe	WriteFile (0×00000248, 0×0158e538, 2048, 0×06c6f820, NULL)
1bc44.exe	SetFilePointerEx (0×00000248, { u = { LowPart = 2429786112, HighPart = 2 }, QuadPart = 11019720704 }, NULL, FILE_BEGIN)
1bc44.exe	WriteFile (0×00000248, 0×0158e538, 2048, 0×06c6f820, NULL)
1bc44.exe	SetFilePointerEx (0×00000248, { u = { LowPart = 4098506752, HighPart = 2 }, QuadPart = 12688441344 }, NULL, FILE_BEGIN)
1bc44.exe	WriteFile (0×00000248, 0×0158e538, 2048, 0×06c6f820, NULL)
1bc44.exe	SetFilePointerEx (0×00000248, { u = { LowPart = 4098508800, HighPart = 2 }, QuadPart = 12688443392 }, NULL, FILE_BEGIN)

Module	API
1bc44.exe	WriteFile (0×00000248, 0×0158e538, 2048, 0×06c6f820, NULL)
1bc44.exe	SetFilePointerEx (0×00000248, { u = { LowPart = 344981504, HighPart = 0 }, QuadPart = 344981504 }, NULL, FILE_BEGIN)
1bc44.exe	WriteFile (0×00000248, 0×0158e538, 2048, 0×06c6f820, NULL)
1bc44.exe	SetFilePointerEx (0×00000248, { u = { LowPart = 344983552, HighPart = 0 }, QuadPart = 344983552 }, NULL, FILE_BEGIN)
1bc44.exe	WriteFile (0×00000248, 0×0158e538, 2048, 0×06c6f820, NULL)
1bc44.exe	SetFilePointerEx (0×00000248, { u = { LowPart = 1048576, HighPart = 0 }, QuadPart = 1048576 }, NULL, FILE_BEGIN)
1bc44.exe	WriteFile (0×00000248, 0×0158e538, 2048, 0×06c6f820, NULL)

Appendix B: YARA Rules

```
import "pe"
import "hash"

rule HermeticWiper{
   meta:
      author = "CNANCE, Insikt Group, Recorded Future"
      date = "2022-02-24"
      description = "Rule to detect HermeticWiper malware"
      version = "1.0"
      hash = "1bc44eef75779e3ca1eefb8ff5a64807dbc942b1e4a2672d77b9f6928d292591"
      hash = "0385eeab00e946a302b24a91dea4187c1210597b8e17cd9e2230450f5ece21da"
      hash = "2c10b2ec0b995b88c27d141d6f7b14d6b8177c52818687e4ff8e6ecf53adf5bf"
      RF_MALWARE = "HermeticWiper"

   strings:
      // paths
      $p1 = "\\\\.\\EPMNTDRV\\%u" fullword wide
      $p2 = "\\\\.\\PhysicalDrive%u" fullword wide

      // disable crash dumps
      $r1 = "SYSTEM\\CurrentControlSet\\Control\\CrashControl" fullword wide
      $r2 = "CrashDumpEnabled" fullword wide

      // privileges
      $s1 = "SeLoadDriverPrivilege" fullword wide
      $s2 = "SeBackupPrivilege" fullword wide
        // stack string: S.eS.hu.td.o..i.vi.le.ge
      $s3 = { c7 4? ?? 61 00 62 00 c7 4? ?? 63 00 64 00 c7 4? ?? 65 00 66 00 c7 4? ?? 67 00 68 00 c7 4? ?? 69 00 6a 00 c7
4? ?? 6b 00 6c 00 c7 4? ?? 6d 00 6e 00 c7 4? ?? 6f 00 70 00 c7 4? ?? 71 00 72 00 c7 4? ?? 73 00 74 00 c7 4? ?? 75 00 76
00 c7 4? ?? 77 00 78 00 c7 4? ?? 79 00 7a 00  }

   condition:
      uint16(0) == 0×5a4d // PE file
      and filesize > 90KB
      and all of them
      and for any i in (0..pe.number_of_signatures): (
         pe.signatures[i].thumbprint == "1ae7556dfacd47d9efbe79be974661a5a6d6d923" // Hermetica Digital Ltd certificate
         )
      and for 2 i in (0..pe.number_of_resources): (
         pe.resources[i].type_string == "R\x00C\x00D\x00A\x00T\x00A\x00"
         and (
               // Check resource names
               (
                  pe.resources[i].name_string == "D\x00R\x00V\x00_\x00X\x006\x004\x00"
                  or pe.resources[i].name_string == "D\x00R\x00V\x00_\x00X\x008\x006\x00"
                  or pe.resources[i].name_string == "D\x00R\x00V\x00_\x00X\x00P\x00_\x00X\x006\x004\x00"
                  or pe.resources[i].name_string == "D\x00R\x00V\x00_\x00X\x00P\x00_\x00X\x008\x006\x00"
               )
```

```
            or
            // Check hashes for EaseUS driver
            (
               hash.sha256(pe.resources[i].offset, pe.resources[i].length) ==
"e5f3ef69a534260e899a36cec459440dc572388defd8f1d98760d31c700f42d5"
               or hash.sha256(pe.resources[i].offset, pe.resources[i].length) ==
"b01e0c6ac0b8bcde145ab7b68cf246deea9402fa7ea3aede7105f7051fe240c1"
               or hash.sha256(pe.resources[i].offset, pe.resources[i].length) ==
"b6f2e008967c5527337448d768f2332d14b92de22a1279fd4d91000bb3d4a0fd"
               or hash.sha256(pe.resources[i].offset, pe.resources[i].length) ==
"fd7eacc2f87aceac865b0aa97a50503d44b799f27737e009f91f3c281233c17d"
            )
         )
      )
}

rule MAL_PartyTicket{
   meta:
      author = "LKAYE, Insikt Group, Recorded Future"
      date = "2022-02-24"
      description = "Rule to detect pseudo-ransomware associated with HermeticWiper malware"
      version = "1.0"
      hash = "4dc13bb83a16d4ff9865a51b3e4d24112327c526c1392e14d56f20d6f4eaf382"
      RF_MALWARE = "HermeticWiper"

   strings:
      $s1 = "403forBiden" ascii
      $s2 = "wHiteHousE" ascii
      $s3 = ".exe.gif.htm.ico.iso.jpg.mp3.msi.odt." ascii //this is fairly unusual for modern, professional ransomware to encrypt
exes
      $s4 = "main.voteFor403" ascii
      $s5 = "main.n1hk9" ascii
      $s6 = "Thank you for your vote!" ascii //part of ransom note
      $s7 = "photoes" ascii //misspelling in ransom note

   condition:
      uint16(0) == 0×5a4d and
      filesize > 3000KB and
      all of them
}
```

Appendix C: IOCs

HermeticWiper Sample (SHA256):

```
1bc44eef75779e3ca1eefb8ff5a64807dbc942b1e4a2672d77b9f6928d292591
0385eeab00e946a302b24a91dea4187c1210597b8e17cd9e2230450f5ece21da
3c557727953a8f6b4788984464fb77741b821991acbf5e746aebdd02615b1767
a64c3e0522fad787b95bfb6a30c3aed1b5786e69e88e023c062ec7e5cebf4d3e
```

Ransomware Sample (SHA256):

```
4dc13bb83a16d4ff9865a51b3e4d24112327c526c1392e14d56f20d6f4eaf382
```

About Recorded Future

Recorded Future is the world's largest intelligence company. The Recorded Future Intelligence Platform provides the most complete coverage across adversaries, infrastructure, and targets. By combining persistent and pervasive automated data collection and analytics with human analysis, Recorded Future provides real-time visibility into the vast digital landscape and empowers clients to take proactive action to disrupt adversaries and keep their people, systems, and infrastructure safe. Headquartered in Boston with offices and employees around the world, Recorded Future works with more than 1,300 businesses and government organizations across 60 countries.

Learn more at recordedfuture.com and follow us on Twitter at @RecordedFuture.