The Anatomy of Wiper Malware, Part 1: Common Techniques | CrowdStrike

By Ioan Iacob - Iulian Madalin Ionita

Archived: 2026-04-05 13:29:00 UTC

This is the first blog post in a four-part series. Read Part 2 | Part 3 | Part 4. This blog post is the first in a four-part series in which CrowdStrike's Endpoint Protection Content Research Team will dive into various wipers discovered by the security community over the past 10 years. Our goal is to review in depth the various techniques employed by wipers that target the Windows operating system.

Background

A wiper is a type of malware with a single purpose: to erase user data beyond recoverability. Wipers are used to destroy computer networks in public or private companies ranging from industrial to entertainment sectors. Threat actors also use wipers to cover up traces left after an intrusion, weakening their victim's ability to respond.

Wipers gained popularity back in 2012, when Saudi Arabia's Saudi Aramco and Qatar's RasGas oil companies were targeted by threat actors using the Shamoon family of wipers. After four years in which little to no wiper activity was observed, the Shamoon wiper resurfaced in 2016 with threat actors having the same goals and targets in mind.

The year 2017 put multiple wiper families on our radar. A wiper variant of Petya was used to target multiple institutions in Ukraine, Russia and western Europe. Institutions in Israel and Germany faced the wipers named SQLShred and Ordinypt, respectively, which masqueraded as ransomware. Middle Eastern companies again found themselves the target of a wiper, this time one named StoneDrill.

Little wiper activity was observed in the following three years. In 2018, South Korea was the host of the Olympic games that were targeted by threat actors using Olympic Destroyer. In late 2019 and early 2020, Dustman and ZeroCleare were used to target organizations in the energy and industrial sectors from the Middle East.

In 2021 threat actors again targeted the Olympics, now hosted in Tokyo, with a wiper named Tokyo Olympic Wiper. In the same year, a pro-Palestinian wiper dubbed IsraBye was used to target Israeli organizations.

Already, 2022 has been the most active year yet for wipers. Ukraine, while fighting Russian forces in traditional warfare, has seen its government institutions targeted by numerous cyberattacks using the wipers CaddyWiper, DoubleZero, DriveSlayer, IsaacWiper, KillDisk and WhisperGate.

Techniques

Over the years, threat actors have used different strategies to achieve their objectives. During that time, we've studied different adversary strategies that use wipers singularly or in combination with other destructive

techniques. While quick and easy techniques can also have quick and easy countermeasures, the more advanced and lengthy ones *may* give victims a chance to react in time but usually not without difficulty.

Ransomware and wipers share some techniques. Both walk the disk in search of files to modify or corrupt, and both are capable of making data recovery impossible for the victim. But in this latter aspect lies one of the biggest differences between the two threats: ransomware typically enables file restoration for victims who pay the ransom, whereas the objective of wipers is to destroy files beyond recoverability. Another difference is in performance; because wipers need not read the data from disk, they work faster and require fewer resources than ransomware.

One of the easiest techniques we've found in the analyzed wiper samples is to merely delete the files on disk. Yet this technique could allow forensics examiners to recover the files by carving them out from the raw disk. Because standard deletion is not a secure method, threat actors have resorted to overwriting target files with bogus data. To increase the speed of the operation, some wipers overwrite only the first part of the target file, while others resort to wiping the Master Boot Record (MBR).

As we'll discuss, these techniques vary in their unique advantages and weaknesses, as well as in the degrees of recoverability of destroyed data. Each of these techniques demands a different course of action to properly detect and respond to the various threats posed by destructive wiper malware families.

File Discovery

In search of files to destroy or corrupt, most wipers recursively iterate through the file system by using Windows APIs like **FindFirstFile** and **FindNextFile**.

```
// e2ecec43da974db02f624ecadc94baf1d21fd1a5c4990c15863bb9929f781a0a
IterateFilesAndWipe(wchar_t *Format)
{
  // ...
  FirstFileW = FindFirstFileW(FileName, &FindFileData);
  // ...
  do
  {
  // ...
    if ( (FindFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY) != 0 )
    {
      // ...
      IterateFilesAndWipe(strfileName);
    }
    else
    { // ...
      WipeFile(strfileName);
    }
  }
  while ( FindNextFileW(FirstFileW, &FindFileData) );
}
```

Figure 1. File iteration via FindFirstFile and FindNextFile APIs

While some wipers immediately overwrite their targets, Apostle, DoubleZero, SQLShred and WhisperGate construct a list of target files to be later processed by the wiping routine. This introduces a bit of overhead before the destructive functionalities are launched, buying the victim time to react.

Wipers are implemented to do as much damage as possible without crashing the operating system. During the file discovery operation, many wipers will implement different strategies to maintain the stability of the operating system. If critical files are overwritten, the machine will crash and the wiper may not achieve the desired outcome. In order to prolong the life of the machine, wipers can delay, skip or prioritize certain targets:

- CaddyWiper, DoubleZero, IsaacWiper, SQLShred, and StoneDrill start the wiping routine with non-OS related drives (including mounted network shares) and directories
- DoubleZero, CaddyWiper, KillDisk, SQLShred, and StoneDrill will skip certain directories (e.g., Windows, Program Files, ProgramData or others) from the wiper routine or delay their destruction at a later time of execution
- KillDisk and WhisperGate skip certain file extensions like DLL, EXE, LIB, SYS
- Ordinypt uses a list of targeted extensions similar to ransomware
- CaddyWiper and SQLShred — if the configuration sets disk destruction — have been observed to first destroy target files and then destroy physical drives or disk volumes

File Overwrite

When it comes to overwriting files, wipers implement different techniques that achieve the same purpose. While some techniques are fairly common, others implement unique methods.

File System API

The standard method to overwrite a file is by using the **CreateFile** and **WriteFile** API combination. The first is used to grab a handle to the desired file and the second is used to overwrite the file contents with new data. This basic technique has been seen in multiple wiper families like CaddyWiper, DoubleZero, IsaacWiper, KillDisk, Meteor (including Stardust and Comet variants), Petya wiper, Shamoon, SQLShred, StoneDrill, and WhisperGate. Some wipers overwrite the entire file — a computationally costly operation — while others only overwrite a fixed number of bytes.

```
// e2ecec43da974db02f624ecadc94baf1d21fd1a5c4990c15863bb9929f781a0a
int WipeFile(LPCWSTR lpFileName)
{
  SetFileAttributesW(lpFileName, FILE_ATTRIBUTE_NORMAL);
  hFile = CreateFileW(
      lpFileName,
      GENERIC_WRITE|GENERIC_READ,
      FILE_ATTRIBUTE_HIDDEN|FILE_ATTRIBUTE_READONLY, 0,
      CREATE_NEW | CREATE_ALWAYS, 0, 0);
  // ...
  FileSize = GetFileSize(hFile, 0);
  hBuff = malloc(FileSize);
  if ( hBuff )
  {
    ExtensionW = PathFindExtensionW(lpFileName);
    if ( SkipTheseExtensions(ExtensionW) )
      WriteFile(hFile, hBuff, FileSize, &lpFileName, 0);
    CloseHandle(hFile);
    free(hBuff);
    return 1;
  }
  return hBuff;
}
```

Figure 2. Determine file size, allocate memory and write to file

In Figure 2, Destover overwrites the entire file by determining its size via **GetFileSize**, allocates memory of the same size, excludes files based on their extension and overwrites the file using **WriteFile**.

File IOCTL

While the previous method was the most common among the researched samples, DoubleZero implements a second mechanism for overwriting files. In order to overwrite the entire file with zeros, this wiper uses the **NtFsControlFile** API to send the **FSCTL_SET_ZERO_DATA** control code to the file system driver along with the size of the file to be overwritten.

```
// 30b3cbe8817ed75d8221059e4be35d5624bd6b5dc921d4991a7adc4c3eb5de4a
SafeFileHandle safeFileHandle = null;
ulong lpFileSize = 0UL;
NtOpenFile(out safeFileHandle,
           GENERIC_READ | GENERIC_WRITE | SYNCHRONIZE,
           ref objectAttributes,
           ref objIoStatusBlock,
           FILE_SHARE_READ | FILE_SHARE_WRITE | FILE_SHARE_DELETE,
           FILE_SYNCHRONOUS_IO_NONALERT);
GetFileSizeEx(safeFileHandle, out lpFileSize);
FILE_ZERO_DATA_INFORMATION inputBufferZeroData = default(FILE_ZERO_DATA_INFORMATION);
inputBufferZeroData.FileOffset = 0;
inputBufferZeroData.BeyondFinalZero = lpFileSize;
try {
    IntPtr inputBufferZeroDataPtr = Marshal.AllocHGlobal(Marshal.SizeOf(inputBufferZeroData));
    Marshal.StructureToPtr(inputBufferZeroData, inputBufferZeroDataPtr, false);
    NtFsControlFile(safeFileHandle, IntPtr.Zero, IntPtr.Zero, IntPtr.Zero,
                    ref objIoStatusBlock, FSCTL_SET_ZERO_DATA,
                    inputBufferZeroDataPtr, Marshal.SizeOf(inputBufferZeroData), IntPtr.Zero, 0);
}
finally {
    CloseHandle(safeFileHandle.DangerousGetHandle());
}
```

Figure 3. DoubleZero uses FCSTL_SET_ZERO_DATA to overwrite file contents

File Deletion

When the operating system deletes a file from disk, the corresponding sectors are not overwritten with “null” data, they are only marked as unused. This indicates that the raw sectors are free to use when other files are created. Ordinypt, Olympic wiper and Apostle wipers implement simple file deletion, where files are only deleted, not overwritten. In this case, the data can still be recovered from the disk via file carving techniques used in digital forensics. To make the files unrecoverable, secure file deletion needs to be implemented and it requires the files to be overwritten before they are deleted from the disk.

Most wipers do not need to delete the files because their contents have been destroyed, but some implement file deletion. This is the case of Destover, KillDisk, Meteor (Stardust/Comet), Shamoon, SQLShred, and StoneDrill which overwrite the target files with random bytes. Only after replacing the file contents, the file is deleted from disk via the **DeleteFile** API.

The following code snippet displays an implementation of **File Deletion** and **File Overwrite** found in the Shamoon wiper.

```
// ...
HANDLE hFile = CreateFileW ( "C:\\Users\\Public\\Downloads\\desktop.ini", ... );
// ...
int iSize = GetFileSize ( hFile,...);
// ...
WriteFile ( hFile, hBuffer, iSize, pNoBytesOW, NULL);
// ...
FlushFileBuffers ( hFile);
// ...
CloseHandle ( hFile);
// ...
DeleteFileW ("C:\\Users\\Public\\Downloads\\desktop.ini");
// ...
```

Figure 4. How Shamoon wiper overwrites and deletes files

Although families like Apostle and Ordinypt do not implement a secure deletion, they are still considered destructive because file carving is not a perfect recovery technique.

Drive Destruction

Some wipers go one step further and attempt to destroy the contents of the disk itself, not just files. This approach provides several advantages to attackers and makes recovery more difficult, if not impossible. Because files may be fragmented across the disk, wiping the files will require the hard disk drive actuator arm to commute to multiple locations, thus decreasing wiping speeds. Overwriting the raw sectors in successive order is advantageous because it drastically increases the speed of the wiping operation. This also applies to modern solid state drives where sequential access is still more performant than random access.

Wiping raw sectors also removes any file system information like partitioning tables, journaling, parity data, metadata and even OS protected files. These operations are equivalent to raw full-disk formatting, ensuring that files cannot be recovered via any forensic methods.

Disk Write

Similar to the way files can be overwritten, IsaacWiper, KillDisk, Petya wiper variant, SQLShred, StoneDrill, WhisperGate and DriveSlayer use the same **CreateFile** and **WriteFile** APIs to overwrite physical disks (\\.\PhysicalDisk0) and/or volumes (\\.\c:) with either random or predefined bytes buffers. “PhysicalDisk0” is used to access the first sector of a disk, where the Master Boot Record (MBR) is stored, while “\\.\C:” will allow the wiper to reference the first sector of the partition.

```
// a196c6b8ffcb97ffb276d04f354696e2391311db3841ae16c8c9f56f36a38e92
// ...
qmemcpy(lpBuffer, pNewMBRData, 0x2000u);
hFile = CreateFileW(
    L"\\\\.\\PhysicalDrive0",
    GENERIC_ALL,
    FILE_SHARE_READ | FILE_SHARE_WRITE,
    0,
    OPEN_EXISTING,
    0, 0);
WriteFile(hFile, lpBuffer, 0x200u, 0, 0);
CloseHandle(hFile);
// ...
```

Figure 5. Overwrite the MBR of the drive 0 via CreateFile and WriteFile APIs

The code snippet displays an implementation found in various wipers to delete the MBR by directly accessing the disk. The MBR is a structure that resides in the first sector of the disk and holds information about how the disk is formatted into one or multiple partitions. Deleting this structure removes information about the partitions making the system unbootable and also the files present in the partitions inaccessible.

Disk Drive IOCTL

Instead of using the **WriteFile** APIs for overwriting the physical disk, CaddyWiper wipes the disk by sending it a Input/Output Control (IOCTL) code. The **IOCTL_DISK_SET_DRIVE_LAYOUT_EX** IOCTL is sent via the **DeviceIoControl** API alongside a buffer filled with zeros in order to wipe information about drive partitions including MBR and/or GUID Partition Table (GPT).

The code snippet below displays the implementation found in CaddyWiper.

```
// a294620543334a721a2ae8eaaf9680a0786f4b9a216d75b55cfd28f39e9430ea
loopCounter = 9;
bytesReturned = 0;
wcscpy(str_physical_drive_w, L"\\\\.\\PHYSICALDRIVE9");
do {
  hDevice = CreateFileW(  str_physical_drive_w,
                          GENERIC_WRITE|GENERIC_READ,
                          FILE_SHARE_READ | FILE_SHARE_WRITE,
                          NULL,
                          OPEN_EXISTING,
                          FILE_ATTRIBUTE_NORMAL,
                          NULL);
  if ( hDevice != INVALID_HANDLE_VALUE ) {
    DeviceIoControl(  hDevice,
                      IOCTL_DISK_SET_DRIVE_LAYOUT_EX,
                      &obj_DRIVE_LAYOUT_INFORMATION_EX ,
                      sizeof(obj_DRIVE_LAYOUT_INFORMATION_EX),
                      NULL,
                      0,
                      &bytesReturned,
                      NULL);
    CloseHandle(hDevice);
  }
  --LOBYTE(str_physical_drive_w[17]);
  result = loopCounter--;
}
while ( result );
```

Figure 6. Wipers corrupt the disk layout using IOCTL_DISK_SET_DRIVE_LAYOUT_EX

File Contents

As discussed previously, wipers may implement destructive actions on the contents of the file to reduce chances of recovery. We observed multiple approaches when deciding the data to be written over the target files. Some samples overwrite the files with the same data across the entire length, others randomize the contents, while others write predefined buffers to the target files.

Overwrite with Same Byte Value

A simple method is to write the same byte over the entire file contents. Wiper families like CaddyWiper, DoubleZero, KillDisk, Meteor (with its Stardust/Comet variants) and SQLShred implement this technique.

This method does not add any overhead to the wiping process, but might leave an opportunity to recover the data via magnetic-force microscopy.

Overwrite with Random Bytes

To avoid any potential weakness of the previous method, threat actors can decide to generate random data to be used while overwriting files. Even some forensic tools implement secure wiping by overwriting the disk or file multiple times with random data, leaving no chance for magnetic-force microscopy to recover the data.

Oftentimes the random buffer is generated via the **seed** and **rand** functions, followed by a write to the file. Generating random data adds overhead, thus lengthening the wiping times. Destover, IsaacWiper, KillDisk, SQLShred and StoneDrill are a few examples of wipers that overwrite target files with random data.

IsaacWiper implements its own pseudorandom number generator to fill a memory buffer, an implementation of Mersenne Twister PRNG.

```
// e2ecec43da974db02f624ecadc94baf1d21fd1a5c4990c15863bb9929f781a0a
int WipeFile(LPCWSTR lpFileName)
{
  SetFileAttributesW(lpFileName, FILE_ATTRIBUTE_NORMAL);
  hFile = CreateFileW(lpFileName, ...);
  // ...
  FileSize = GetFileSize(hFile, 0);
  hBuff = malloc(FileSize);
  if ( hBuff )
  {
    ExtensionW = PathFindExtensionW(lpFileName);
    if ( SkipTheseExtensions(ExtensionW) )
      WriteFile(hFile, hBuff, FileSize, &lpFileName, 0);
    CloseHandle(hFile);
    free(hBuff);
    return 1;
  }
  return hBuff;
}
```

Figure 7. Malloc is used to “generate random” bytes that will be written to the file

In Figure 6, Destover takes advantage of a caveat in the **malloc** function to generate “random” data. **Malloc** will allocate a memory buffer, but it will contain residual data from previous usage of that memory page that is then written over the entire length of the file.

Overwrite with Predefined Data

The final method to discuss is the use of hard coded data to overwrite files. This method eliminates the overhead introduced by generating random bytes, thus increasing the speed of data destruction.

Shamoon overwrites files with a predefined jpeg image that is hardcoded and obfuscated in the wiper binary. It uses the **WriteFile** API to write the image; the header of the jpeg is seen in the memory view in the second half of the screenshot.

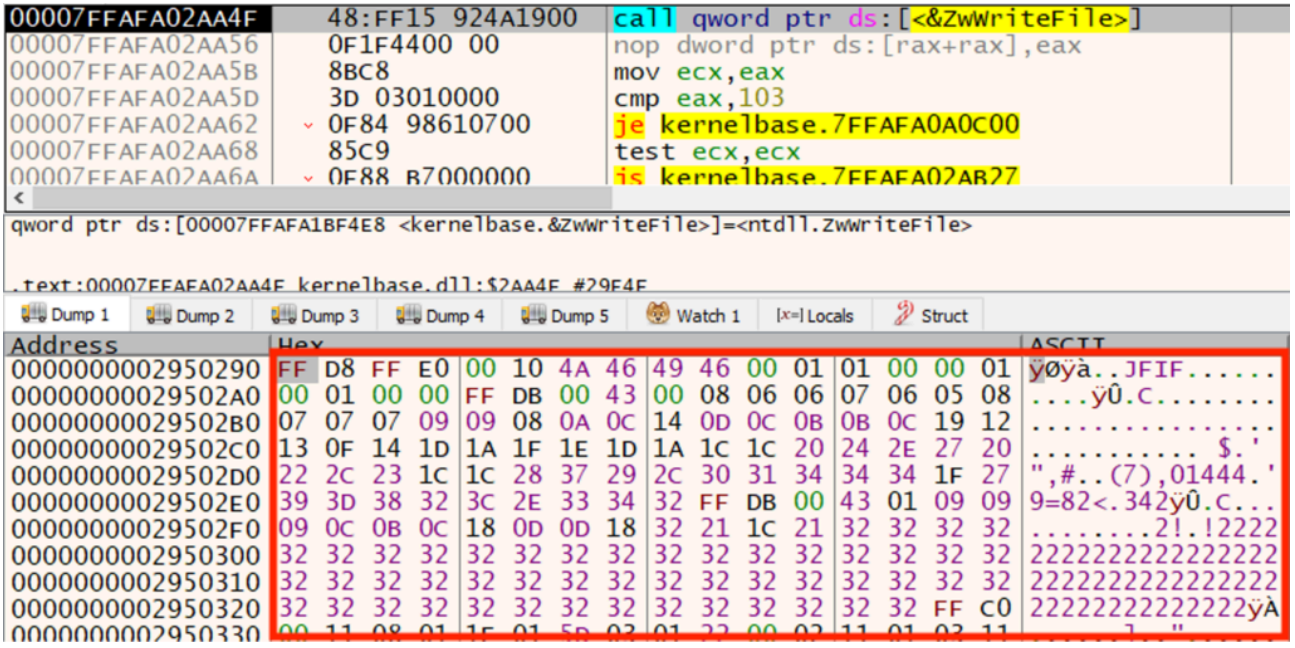

Figure 8. Debugger view, showcasing the JPEG image being written to a file

In contrast, the wiper IsraBye writes only a message to the file contents, and it does not overwrite every byte in the file content, leaving some data available for forensics analysts to extract. However, even though it is not as destructive as others, this wiper is able to overwrite the file header, reducing the possibility of data carving or recovery.

```
// 5a209e40e0659b40d3d20899c00757fa33dc00ddcac38a3c8df004ab9051de0d
this.content = "Custom message";
// ...
string[] files = Directory.GetFiles(path);
int totalNumberOfFiles = files.Length - 1;
int index = 0;
for (;;) {
    if (index > totalNumberOfFiles)
        break;
    File.WriteAllText(files[index], this.content);
    File.Move(files[index], files[index] + ".israbye");
    index++;
}
```

Figure 9. IsraBye code snippet used to file overwrite and file rename

How the CrowdStrike Falcon® Platform Protects Customers Against Wipers

The CrowdStrike Falcon® platform takes a layered approach to protect workloads. Using on-sensor and cloud-based machine learning, behavior-based detection using indicators of attack (IOAs), and intelligence related to tactics, techniques and procedures (TTPs) employed by threat actors, the Falcon platform equips users with visibility, threat detection, automated protection and continuous monitoring to rapidly detect and mitigate threats in any environment.

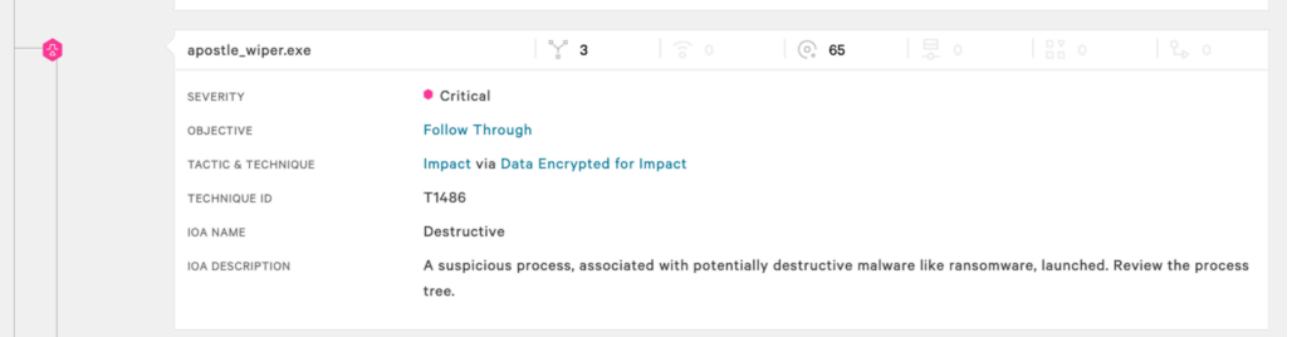

Figure 10. Falcon UI screenshot showcasing detection of Apostle by Falcon sensor.

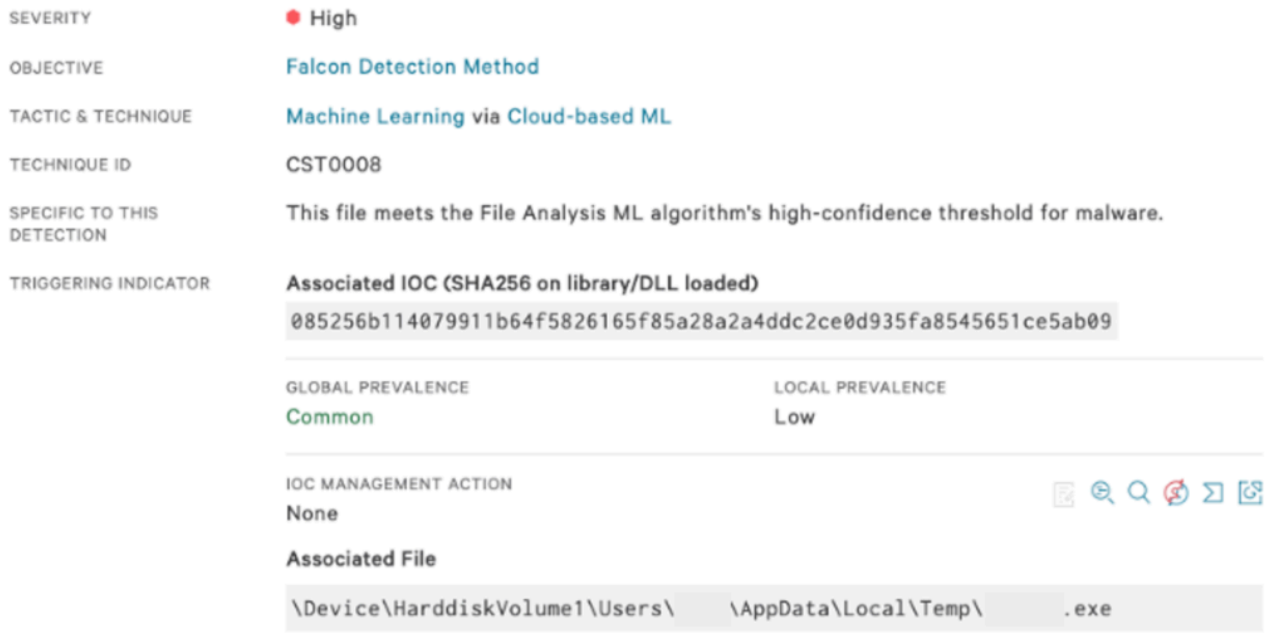

Figure 11. Falcon UI screenshot showcasing detection of Ordinypt by Falcon sensor.

Summary

Depending on the skill set of different threat actors, wipers have implemented different techniques in order to sabotage the operations of their targets. Most often, wipers use file system specific APIs to iterate through files and overwrite and delete as many as possible.

Some wipers don’t target only the files from the victim’s machine, but may also target the raw disk. This latter technique provides several advantages like increased wiping speeds for example. Also, it may bypass security

measures implemented by the file system or operating system and may even be invisible to security products.

To further increase the speed of the operations, some wipers do not overwrite the entire length of the target data, but only parts of it enough to make the files unrecoverable. To increase the destruction capability, randomizing the contents overwritten to the files seems like a good approach, but it becomes a time intensive task. An interesting and time efficient approach seen in some wipers is the usage of **malloc** to use garbage data to overwrite the target.

In part two of this wiper series, we will dive into how wipers use legitimate third-party drivers to destroy files as well as disk clusters.

Hashes

Wiper name	**SHA256 hash value**
Apostle	6fb07a9855edc862e59145aed973de9d459a6f45f17a8e779b95d4c55502dcce 19dbed996b1a814658bef433bad62b03e5c59c2bf2351b793d1a5d4a5216d27e
CaddyWiper	a294620543334a721a2ae8eaaf9680a0786f4b9a216d75b55cfd28f39e9430ea
Destover	e2ecec43da974db02f624ecadc94baf1d21fd1a5c4990c15863bb9929f781a0a
DoubleZero	3b2e708eaa4744c76a633391cf2c983f4a098b46436525619e5ea44e105355fe 30b3cbe8817ed75d8221059e4be35d5624bd6b5dc921d4991a7adc4c3eb5de4a
DriveSlayer	0385eeab00e946a302b24a91dea4187c1210597b8e17cd9e2230450f5ece21da 1bc44eef75779e3ca1eefb8ff5a64807dbc942b1e4a2672d77b9f6928d292591 a259e9b0acf375a8bef8dbc27a8a1996ee02a56889cba07ef58c49185ab033ec
Dustman	f07b0c79a8c88a5760847226af277cf34ab5508394a58820db4db5a8d0340fc7
IsaacWiper	13037b749aa4b1eda538fda26d6ac41c8f7b1d02d83f47b0d187dd645154e033 7bcd4ec18fc4a56db30e0aaebd44e2988f98f7b5d8c14f6689f650b4f11e16c0
IsraBye	5a209e40e0659b40d3d20899c00757fa33dc00ddcac38a3c8df004ab9051de0d
KillDisk	8a81a1d0fae933862b51f63064069aa5af3854763f5edc29c997964de5e284e5 1a09b182c63207aa6988b064ec0ee811c173724c33cf6dfe36437427a5c23446
Meteor and Comet/Stardust	2aa6e42cb33ec3c132ffce425a92dfdb5e29d8ac112631aec068c8a78314d49b d71cc6337efb5cbbb400d57c8fdeb48d7af12a292fa87a55e8705d18b09f516e 6709d332fbd5cde1d8e5b0373b6ff70c85fee73bd911ab3f1232bb5db9242dd4 9b0f724459637cec5e9576c8332bca16abda6ac3fbbde6f7956bc3a97a423473
Ordinypt	085256b114079911b64f5826165f85a28a2a4ddc2ce0d935fa8545651ce5ab09
Petya	0f732bc1ed57a052fecd19ad98428eb8cc42e6a53af86d465b004994342a2366 fd67136d8138fb71c8e9677f75e8b02f6734d72f66b065fc609ae2b3180a1cbf 4c1dc737915d76b7ce579abddaba74ead6fdb5b519a1ea45308b8c49b950655c

Shamoon	e2ecec43da974db02f624ecadc94baf1d21fd1a5c4990c15863bb9929f781a0a c7fc1f9c2bed748b50a599ee2fa609eb7c9ddaeb9cd16633ba0d10cf66891d8a 7dad0b3b3b7dd72490d3f56f0a0b1403844bb05ce2499ef98a28684fbccc07b4 8e9681d9dbfb4c564c44e3315c8efb7f7d6919aa28fcf967750a03875e216c79 f9d94c5de86aa170384f1e2e71d95ec373536899cb7985633d3ecfdb67af0f72 4f02a9fcd2deb3936ede8ff009bd08662bdb1f365c0f4a78b3757a98c2f40400
SQLShred/Agrius	18c92f23b646eb85d67a890296000212091f930b1fe9e92033f123be3581a90f e37bfad12d44a247ac99fdf30f5ac40a0448a097e36f3dbba532688b5678ad13
StoneDrill	62aabce7a5741a9270cddac49cd1d715305c1d0505e620bbeaec6ff9b6fd0260 2bab3716a1f19879ca2e6d98c518debb107e0ed8e1534241f7769193807aac83 bf79622491dc5d572b4cfb7feced055120138df94ffd2b48ca629bb0a77514cc
Tokyo Olympic wiper	fb80dab592c5b2a1dcaaf69981c6d4ee7dbf6c1f25247e2ab648d4d0dc115a97 c58940e47f74769b425de431fd74357c8de0cf9f979d82d37cdcf42fcaaeac32
WhisperGate	a196c6b8ffcb97ffb276d04f354696e2391311db3841ae16c8c9f56f36a38e92 44ffe353e01d6b894dc7ebe686791aa87fc9c7fd88535acc274f61c2cf74f5b8 dcbbae5a1c61dbbbb7dcd6dc5dd1eb1169f5329958d38b58c3fd9384081c9b78
ZeroCleare	becb74a8a71a324c78625aa589e77631633d0f15af1473dfe34eca06e7ec6b86

Additional Resources

- *Find out more about today's adversaries and how to combat them at Fal.Con 2022, the cybersecurity industry's most anticipated annual event. Register now and meet us in Las Vegas, Sept. 19-21!*
- *Learn how the powerful CrowdStrike Falcon® platform provides comprehensive protection across your organization, workers and data, wherever they are located.*
- *Get a full-featured free trial of CrowdStrike Falcon® Prevent™ and see for yourself how true next-gen AV performs against today's most sophisticated threats.*

Source: https://www.crowdstrike.com/blog/the-anatomy-of-wiper-malware-part-1/