Ransomware Analysis "payload.exe"

November, 2017

Panda Security

Ransomware from the Crysis/Dharma family 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

1. SUMMARY.

The present document compiles the analysis of a ransomware from the Crysis/Dharma family. This ransomware is an evolution of this family, and has been circulating "in the wild" since the end of August. It is an extremely dangerous ransomware, since it encrypts all files located on the local drives as well as shared network directories. It also deletes all Shadow Copies so that the user cannot restore them.

2. INFORMATION ON THE SAMPLE.

MD5	6c8f32b51f3dbdf0ea32dfef7233df32
Size	338.432 bytes
Internal Date	28/10/2017 14:08:00

3. CHARACTERISTICS.

The sample is encrypted and protected by a “packer” that is used to package other malware samples. In order to analyze it, every last layer must be extracted.

Once the final layer is reached, we find a reference to the filename “payload.exe”.

Over the course of the analysis, we discovered that “payload.exe” has most of its chains encrypted. It resolves system calls dynamically.

3.1 Primary actions of the malicious code.

- After executing, it proceeds to resolve the APIs that it will use, so initially no suspicious data import is seen:

```
HMODULE __cdecl sub_4065E0(CHAR *a1)
{
  HMODULE result; // eax@1
  const CHAR *lpLibFileName; // [esp+0h] [ebp-Ch]@1
  LPCSTR lpLibFileNamea; // [esp+0h] [ebp-Ch]@6
  int v4; // [esp+4h] [ebp-8h]@1
  HMODULE hModule; // [esp+8h] [ebp-4h]@3

  result = (HMODULE)a1;
  lpLibFileName = a1;
  v4 = 0;
  while ( *lpLibFileName )
  {
    result = LoadLibraryA(lpLibFileName);
    hModule = result;
    if ( !result )
      break;
    while ( *lpLibFileName )
      ++lpLibFileName;
    for ( lpLibFileNamea = lpLibFileName + 1; ; ++lpLibFileNamea )
    {
      result = (HMODULE)*lpLibFileNamea;
      if ( !*lpLibFileNamea )
        break;
      *(&off_4186B8 + v4++) = GetProcAddress(hModule, lpLibFileNamea);
      while ( *lpLibFileNamea )
        ++lpLibFileNamea;
    }
    while ( *lpLibFileNamea )
      result = (HMODULE)(lpLibFileNamea++ + 1);
    lpLibFileName = lpLibFileNamea + 1;
  }
  return result;
}
```

- It then checks the existence of the mutex called “Global\syncronize_XXXXXXX”. If the mutex does not exist, it takes no action and we get:

```
ext:004081F3 83 C4 10
ext:004081F6 85 C0
ext:004081F8 7E 29
ext:004081FA 8B 45 F8
ext:004081FD 50
ext:004081FE 6A 00
ext:00408200 68 00 00 10 00
ext:00408205 E8 36 E6 FF FF
ext:0040820A 89 45 E0
ext:0040820D 83 7D E0 00
ext:00408211 75 10
ext:00408213 8B 4D F8
ext:00408216 51
ext:00408217 6A 00
ext:00408219 6A 00
ext:0040821B E8 50 E5 FF FF
ext:00408220 89 45 E0
```

```
add     esp, 10h
test    eax, eax
jle     short loc_408223
mov     eax, [ebp+lpName] ; <Global\syncronize_754020A>
push    eax             ; lpName
push    0               ; bInheritHandle
push    100000h         ; dwDesiredAccess
call    OpenMutexW
mov     [ebp+var_20], eax
cmp     [ebp+var_20], 0
jnz     short loc_408223
mov     ecx, [ebp+lpName]
push    ecx             ; lpName
push    0               ; bInitialOwner
push    0               ; lpMutexAttributes
call    CreateMutexW
mov     [ebp+var_20], eax
```

- It decrypts its own code in a list of extensions:

```
call    DecodeaToNewBuffer1
add     esp, 10h
mov     [ebp+lpString], eax ;  <doc(.doc;.docx;.pdf;.xls;.xlsx;.ppt;)arc(.zip;.rar;.bz2;.7z;)dbf(.dbf;)1c8(.1cd;)jpg(.jpg;)>
mov     ecx, [ebp+lpString]
```

- It looks for running services and a few programs, and if it finds them it stops them and kills associated processes:

```
call    DecodeaToNewBuffer1
add     esp, 10h
mov     [ebp+lpString], eax ; 1c8.exe;1cv77.exe;outlook.exe;postgres.exe;mysqld-nt.exe;mysqld.exe;sqlservr.exe;>
push    2
push    80h
push    offset unk_40E080
push    offset unk_4179BC
call    DecodeaToNewBuffer1
add     esp, 10h
mov     [ebp+var_4], eax ; <FirebirdGuardianDefaultInstance;FirebirdServerDefaultInstance;sqlwriter;mssqlserver;sqlserveradhelper;>,0
```

- It creates persistence in the record and in the Start folder:

```
text:004079C4 E8 77 98 FF FF
text:004079C9 83 C4 18
text:004079CC 85 C0
text:004079CE 7E 3C
text:004079D0 68 FF 7F 00 00
text:004079D5 8B 45 E8
text:004079D8 50
text:004079D9 E8 72 F9 FF FF
text:004079DE 83 C4 08
text:004079E1 85 C0
text:004079E3 7E 27
text:004079E5 8B 4D E8
text:004079E8 51
text:004079E9 8B 55 E4
text:004079EC 52
text:004079ED E8 1E FD FF FF
text:004079F2 83 C4 08
text:004079F5 85 C0
text:004079F7 74 13
text:004079F9 8B 45 E8
text:004079FC 50
text:004079FD 8B 4D F0
text:00407A00 51
text:00407A01 E8 0A FE FF FF
```

```
call    AppendText
add     esp, 18h
test    eax, eax
jle     short loc_407A0C
push    7FFFh
mov     eax, [ebp+var_18]
push    eax             ; %windir%\System32\payload.exe_
call    sub_407350
add     esp, 8
test    eax, eax
jle     short loc_407A0C
mov     ecx, [ebp+var_18]
push    ecx
mov     edx, [ebp+var_1C]
push    edx
call    CreaArchivo
add     esp, 8
test    eax, eax
jz      short loc_407A0C
mov     eax, [ebp+var_18]
push    eax
mov     ecx, [ebp+var_10]
push    ecx
call    GeneraPersistenciaRegistro ; Software\Microsoft\Windows\CurrentVersion\Run
```

```
text:00407A8F 83 C4 18
text:00407A92 85 C0
text:00407A94 7E 32
text:00407A96 68 FF 7F 00 00
text:00407A9B 8B 4D E8
text:00407A9E 51
text:00407A9F E8 AC F8 FF FF
text:00407AA4 83 C4 08
text:00407AA7 85 C0
text:00407AA9 74 1D
text:00407AAB 8B 55 E8
text:00407AAE 52
text:00407AAF 8B 45 E4
text:00407AB2 50
text:00407AB3 E8 58 FC FF FF
```

```
add     esp, 18h
test    eax, eax
jle     short loc_407AC8
push    7FFFh
mov     ecx, [ebp+var_18]
push    ecx             ; <%sh(Startup)%\payload.exe_
call    sub_407350
add     esp, 8
test    eax, eax
jz      short loc_407AC8
mov     edx, [ebp+var_18]
push    edx
mov     eax, [ebp+var_1C]
push    eax
call    CreaArchivo
```

- **It eliminates Shadow Copies** using the following command that runs via the "cmd.exe" associated to a named pipe, making it so that it does not need to generate .bat files:

```
text:0040A80A E8 41 72 FF FF                    call    DecodeaToNewBuffer1
text:0040A80F 83 C4 10                          add     esp, 10h
text:0040A812 89 45 90                          mov     [ebp+lpBuffer], eax ; mode con cp select=1251',0Ah
text:0040A812                                               ; vssadmin delete shadows /all /quiet',0Ah
text:0040A812                                               ; Exit',0Ah,0
```

```
if ( CreatePipe(&hReadPipe, &hFile, &PipeAttributes, 0) )
{
  if ( CreatePipe(&v14, &hWritePipe, &PipeAttributes, 0) )
  {
    SetHandleInformation(hFile, 1u, 0);
    SetHandleInformation(v14, 1u, 0);
    StartupInfo.cb = 68;
    StartupInfo.dwFlags = 257;
    StartupInfo.hStdInput = hReadPipe;
    StartupInfo.hStdOutput = hWritePipe;
    StartupInfo.hStdError = hWritePipe;
    StartupInfo.wShowWindow = 0;
    if ( CreateProcessW(lpApplicationName, 0, 0, 0, 1, 0, 0, 0, &StartupInfo, (LPPROCESS_INFORMATION)&hObject) )
    {
      v0 = lstrlen(lpBuffer);
      WriteFile(hFile, lpBuffer, v0, &NumberOfBytesWritten, 0);
      CloseHandle_0(hObject);
      CloseHandle_0(v13);
    }
  }
}
```

- It then initiates various threads looking for the abovementioned processes. It kills these threads in case they activate while the ransomware is performing its encryption.

```
{
  _DWORD *v3; // eax@1

  CreateThread(0, 0, ThreadBuscaYMataProceso, 0, 0, 0);
  v3 = (_DWORD *)AlojaMemoria(12);
  v3[2] = a3;
  v3[1] = a2;
  *v3 = a1;
  return CreateThread(0, 0, ThreadEjecuta, v3, 0, 0);
}
```

- Finally, it initiates threads to create a list of encrypted files:

```
{
  _DWORD *v2; // eax@1

  v2 = (_DWORD *)AlojaMemoria(8);
  *v2 = a1;
  v2[1] = a2;
  return CreateThread(0, 0, ThreadGetLogicalDrives, v2, 0, 0);
}
```

- Within this thread, we find code related to the list of resources located on the network:

```
Enum = 0;
lpBuffer = (struct _NETRESOURCEW *)AlojaMemoria(4096);
BufferSize = 0;
cCount = 0;
if ( !lpNetResource && !WNetOpenEnumW(3u, 1u, 0, 0, &hEnum) )

  BufferSize = 4096;
  for ( cCount = 128; !WNetEnumResourceW(hEnum, &cCount, lpBuffer, &BufferSize); cCount = 128 )
  {
    for ( i = 0; i < cCount; ++i )
    {
      if ( lpBuffer[i].dwType == 1 )
      {
        v4 = lstrlenW(lpString);
        if ( sub_406D50(lpBuffer[i].lpRemoteName, lpString, v4) )
          sub_4093B0(lpBuffer[i].lpRemoteName, a3, 0, a4);
        sub_4093B0(lpBuffer[i].lpRemoteName, a3, 1, a4);
      }
      if ( lpBuffer[i].dwUsage & 2 )
        EnumeraWMNET(&lpBuffer[i], lpString, a3, a4);
    }
    BufferSize = 4096;
  }
  WNetCloseEnum(hEnum);

result = WNetOpenEnumW(2u, 1u, 0, lpNetResource, &hEnum);
if ( !result )
```

This indicates that the ransomware en**crypts files in network drives**. If there is any open access shared file, it will be fully encrypted.

4. PROPAGATION MECHANISM

No mechanism has been detected allowing the propagation of this harmful code to other devices; it does not exploit vulnerabilities in remote systems or attack the credentials of other devices/services.

The malware behaves basically like a “Trojan-ransomware”, meaning human intervention is necessary for the activation of its malicious code (manual execution).

5. INITIAL INFECTION VECTOR

Thanks to the shared intelligence systems of Panda Security, it has been determined that the initial infection vector for distributing this type of malware is usually the RPD (Remote Desktop Protocol).

In such cases, the attackers, using specially prepared tools, violate the device’s Internet-facing credentials to access systems and execute code (in this case, ransomware).

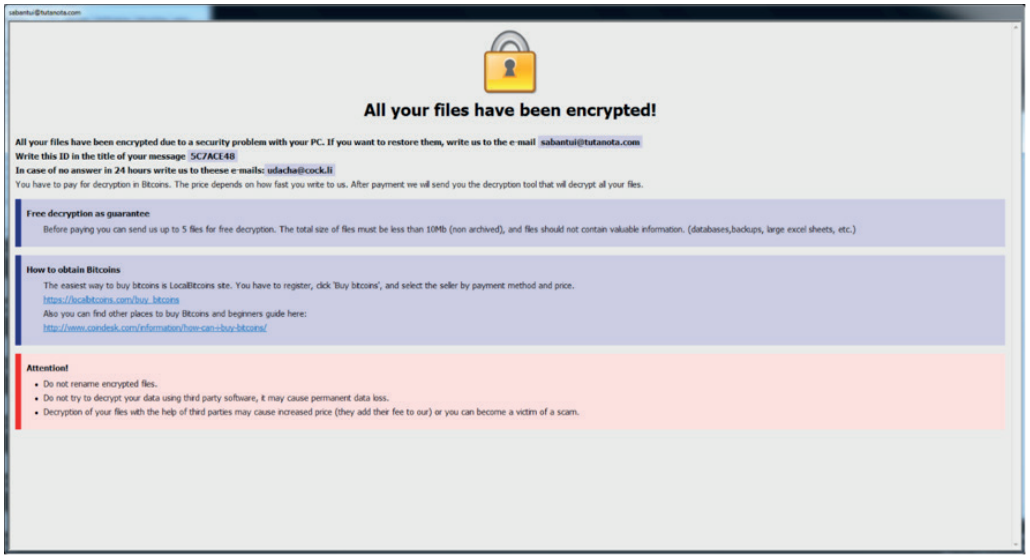
The attacks, therefore, are carried out manually, and the computer/network is considered to have been “hacked”.

6. COMMAND AND CONTROL SERVERS

In this case, no malware communicating autonomously with the command and control server has been detected.

7. RECOVERING FILES

In this case, the authors of this malware chose to provide the user **with two email addresses** to contact if they wish to recover their encrypted files.



TAs shown in the screenshot, the user can send up to 5 files (maximum 10 MB) to be decrypted at no cost. The following code must be indicated in the email subject:

> Write this ID in the title of your message 5C7ACE48

It does refer to the bitcoin cryptocurrency as the means of payment, but does not specify the amount, nor the address (wallet) to which the transfer should be made.

It is understood that once the payment is made, the user will receive a tool to decrypt the files.

Because the information is available from users who have already paid, instead of sending the tool the cybercriminals ask for more money, so paying is risky.

8. PERSISTENCE IN THE SYSTEM

The persistence of the malware is achieved via registration key, as we can see in the following screenshot:

```
text:004079C4 E8 77 98 FF FF      call    AppendText
text:004079C9 83 C4 18            add     esp, 18h
text:004079CC 85 C0               test    eax, eax
text:004079CE 7E 3C               jle     short loc_407A0C
text:004079D0 68 FF 7F 00 00      push    7FFFh
text:004079D5 8B 45 E8            mov     eax, [ebp+var_18]
text:004079D8 50                  push    eax             ; %windir%\System32\payload.exe_
text:004079D9 E8 72 F9 FF FF      call    sub_407350
text:004079DE 83 C4 08            add     esp, 8
text:004079E1 85 C0               test    eax, eax
text:004079E3 7E 27               jle     short loc_407A0C
text:004079E5 8B 4D E8            mov     ecx, [ebp+var_18]
text:004079E8 51                  push    ecx
text:004079E9 8B 55 E4            mov     edx, [ebp+var_1C]
text:004079EC 52                  push    edx
text:004079ED E8 1E FD FF FF      call    CreaArchivo
text:004079F2 83 C4 08            add     esp, 8
text:004079F5 85 C0               test    eax, eax
text:004079F7 74 13               jz      short loc_407A0C
text:004079F9 8B 45 E8            mov     eax, [ebp+var_18]
text:004079FC 50                  push    eax
text:004079FD 8B 4D F0            mov     ecx, [ebp+var_10]
text:00407A00 51                  push    ecx
text:00407A01 E8 0A FE FF FF      call    GeneraPersistenciaRegistro ; Software\Microsoft\Windows\CurrentVersion\Run
```

9. FILE ENCRYPTION

We were able to verify that the file encryption is symmetric, and the algorithm used is AES.

Here we can see the list of file extensions that the attackers are interested in:

```
call    DecodeaToNewBuffer1
add     esp, 10h
mov     [ebp+lpString], eax ;  <doc(.doc;.docx;.pdf;.xls;.xlsx;.ppt;)arc(.zip;.rar;.bz2;.7z;)dbf(.dbf;)1c8(.1cd;)jpg(.jpg;)>
mov     ecx, [ebp+lpString]
```

UnOnce the file encryption thread is initiated, it runs through each and every folder in the system and encrypts all files with the indicated extensions.

It also proceeds to create a list of network folders accessible to perform this same encryption.

The encrypted files are renamed thus:

<Original_name>.id-5C7ACE48.[sabantui@tutanota.com].arena

In each folder where files have been encrypted, a file named “FILES ENCRYPTED.txt” with the following content will be created:

> “all your data has been locked us You want to return?
> write email sabantui@tutanota.com or udacha@cock.li”

10.RECOMMENDATIONS

- Install Adaptive Defense on all network devices and servers. It is advisable to have the Lock Mode activated.
- Upgrade systems and applications to their latest versions to avoid exploitation of vulnerabilities.
- Change systems and administrator passwords to make them more robust.
- Control RDP access on devices that have this option activated.
- Perform general hardening of patches and improve security policies.
- Fortify access from secondary networks.