

SIN CLASIFICAR



Informe Código Dañino CCN-CERT ID-24/16

Ransom.CryptXXX

Septiembre 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
3.2 EXTENSIÓN AÑADIDA A LOS ARCHIVOS CIFRADOS	8
3.2.1 PRIMERA Y SEGUNDA VERSIÓN	8
3.2.2 TERCERA VERSIÓN	9
3.3 ARCHIVOS DE RESCATE	9
3.3.1 PRIMERA Y SEGUNDA VERSIÓN	9
3.3.2 TERCERA VERSIÓN	10
4. CARACTERÍSTICAS DEL CÓDIGO DAÑINO	13
5. DETALLES GENERALES	14
6. PROCEDIMIENTO DE INFECCIÓN	14
7. CARACTERÍSTICAS TÉCNICAS	15
7.1 EXPORTACIÓN XXS0S	16
7.2 EXPORTACIÓN XXS1S	17
7.3 EXPORTACIÓN XXS2S	21
7.4 EXPORTACIÓN XXS3S	21
7.5 EXPORTACIÓN XXS4S	21
8. CIFRADO Y OFUSCACIÓN	22
8.1 CIFRADO	22
8.2 OFUSCACIÓN	22
9. PERSISTENCIA EN EL SISTEMA	23
10. CONEXIONES DE RED	23
11. ARCHIVOS RELACIONADOS	24
12. DETECCIÓN	25
12.1 PRIMERA Y SEGUNDA VERSIÓN	25
12.2 TERCERA VERSIÓN	26
12.3 MANDIANT	27
13. DESINFECCIÓN	27
13.1 PRIMERA Y SEGUNDA VERSIÓN	27

13.1.1 KASPERSKY RANNOH DECRYPTOR	27
13.2 TERCERA VERSIÓN	28
14. INFORMACIÓN DEL ATACANTE	29
14.1 188.0.236.7	29
14.1.1 GEOLOCALIZACIÓN	30
15. REFERENCIAS	30
16. REGLAS DE DETECCIÓN	30
16.1 INDICADOR DE COMPROMISO – IOC	30
16.2 YARA	32

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 "**Ransom.CryptXXX**", el cual ha sido diseñado para instalarse en el sistema, comunicarse con un dominio de Internet, cifrar ciertos archivos y extorsionar a la víctima mostrando una notificación sobre el procedimiento de pago para rescatar los archivos cifrados.

El código dañino se distribuye mayoritariamente mediante *downloaders* que son los encargados de bajar el código dañino de Internet y ejecutarlo como, por ejemplo, *Bedet*. Otro medio de distribución del código dañino es utilizar servidores comprometidos en los cuales se alojó un "*Exploit Kit*" que aprovecha alguna vulnerabilidad en los navegadores de los visitantes de las páginas web alojadas para descargar el código dañino y ejecutarlo en sus sistemas.

3. INFORMACIÓN DE VERSIONES DEL CÓDIGO DAÑINO

El código dañino consta de tres versiones hasta la fecha (agosto de 2016). De las tres versiones existentes solo las dos primeras pueden ser descifradas por herramientas gratuitas de terceros.

3.1 EXTENSIONES A CIFRAR

El código dañino cifra los archivos con las siguientes extensiones:

.3DM	.CSS	.FL	.KIC	.OTS	.RIS	.TMX
.3DS	.CSV	.FLA	.KLG	.OTT	.RIX	.TN
.3G2	.CSY	.FLI	.KML	.OVP	.RLE	.TNE
.3GP	.CUE	.FLR	.KMZ	.OVR	.RLI	.TPC
.4DB	.CV5	.FLV	.KNT	.OWC	.RM	.TPI
.4DL	.CVG	.FM5	.KON	.OWG	.RNG	.TRM
.4MP	.CVI	.FMV	.KPG	.OYX	.RPD	.TVJ
.7Z	.CVS	.FODT	.KWD	.OZB	.RPF	.TXT
.A3D	.CVX	.FOL	.LAY	.OZJ	.RPT	.U3D
.ABM	.CWT	.FP3	.LAY6	.OZT	.RRI	.U3I
.ABS	.CXF	.FP4	.LBM	.P12	.RSB	.UDB
.ABW	.CYI	.FP5	.LBT	.P7S	.RSD	.UFO
.ACCDDB	.DAD	.FP7	.LDF	.P96	.RSR	.UFR
.ACT	.DAF	.FPOS	.LGC	.P97	.RSS	.UGA
.ADN	.DB	.FPT	.LIS	.PAGES	.RST	.UNX
.ADP	.DB3	.FPX	.LIT	.PAL	.RT	.UOF
.AES	.DBF	.FRM	.LJP	.PAN	.RTD	.UOP
.AF2	.DBK	.FRT	.LMK	.PANO	.RTF	.UOT
.AF3	.DBT	.FT10	.LNT	.PAP	.RTX	.UPD
.AFT	.DBV	.FT11	.LP2	.PAQ	.RUN	.USR
.AFX	.DBX	.FT7	.LRC	.PAS	.RW2	.UTF8
.AGIF	.DCA	.FT8	.LST	.PB	.RWL	.UTXT
.AGP	.DCB	.FT9	.LTR	.PBM	.RZK	.V12
.AHD	.DCH	.FTN	.LTX	.PC1	.RZN	.VB
.AI	.DCS	.FWDN	.LUA	.PC2	.S2MV	.VBR
.AIC	.DCT	.FXC	.LUE	.PC3	.S3M	.VBS
.AIF	.DCU	.FXG	.LUF	.PCD	.SAF	.VCF
.AIM	.DCX	.FZB	.LWO	.PCS	.SAI	.VCT
.ALBM	.DDL	.FZV	.LWP	.PCT	.SAM	.VCXPROJ
.ALF	.DDOC	.GADGET	.LWS	.PCX	.SAVE	.VDA
.ANI	.DDS	.GBK	.LYT	.PDB	.SBF	.VDB

.ANS	.DED	.GBR	.LYX	.PDD	.SCAD	.VDI
.APD	.DF1	.GCDP	.M	.PDF	.SCC	.VEC
.APK	.DG	.GDB	.M3D	.PDM	.SCH	.VFF
.APM	.DGN	.GDOC	.M3U	.PDN	.SCI	.VMDK
.APNG	.DGS	.GED	.M4A	.PDS	.SCM	.VML
.APP	.DHS	.GEM	.M4V	.PDT	.SCT	.VMX
.APS	.DIB	.GEO	.MA	.PE4	.SCV	.VNT
.APT	.DIF	.GFB	.MAC	.PEF	.SCW	.VOB
.APX	.DIP	.GGR	.MAN	.PEM	.SDB	.VPD
.ARC	.DIZ	.GIF	.MAP	.PFF	.SDF	.VPE
.ART	.DJV	.GIH	.MAQ	.PFI	.SDM	.VRML
.ARW	.DJVU	.GIM	.MAT	.PFS	.SDOC	.VRP
.ASC	.DM3	.GIO	.MAX	.PFV	.SDW	.VSD
.ASE	.DMI	.GLOX	.MB	.PFX	.SEP	.VSDM
.ASF	.DMO	.GPD	.MBM	.PGF	.SFC	.VSDX
.ASK	.DNC	.GPG	.MBOX	.PGM	.SFW	.VSM
.ASM	.DNE	.GPN	.MDB	.PHM	.SGM	.VST
.ASP	.DOC	.GPX	.MDF	.PHP	.SH	.VSTX
.ASPX	.DOCB	.GRO	.MDN	.PI1	.SIG	.VUE
.ASW	.DOCM	.GROB	.MDT	.PI2	.SITX	.VW
.ASX	.DOCX	.GRS	.ME	.PI3	.SK1	.WAV
.ASY	.DOCZ	.GSD	.MEF	.PIC	.SK2	.WB1
.ATY	.DOT	.GTHR	.MELL	.PICT	.SKM	.WBC
.AVI	.DOTM	.GTP	.MFD	.PIF	.SLA	.WBD
.AWDB	.DOTX	.GV	.MFT	.PIX	.SLD	.WBK
.AWP	.DP1	.GWI	.MGCB	.PJPG	.SLDX	.WBM
.AWT	.DPP	.GZ	.MGMT	.PJT	.SLK	.WBMP
.AWW	.DPX	.H	.MGMX	.PL	.SLN	.WBZ
.AZZ	.DQY	.HBK	.MID	.PLT	.SLS	.WCF
.BAD	.DRW	.HDB	.MIN	.PLUGIN	.SMF	.WDB
.BAY	.DRZ	.HDP	.MKV	.PM	.SMIL	.WDP
.BBS	.DSK	.HDR	.MMAT	.PMG	.SMS	.WEBP
.BDB	.DSN	.HHT	.MML	.PNG	.SOB	.WGZ
.BDP	.DSV	.HIS	.MNG	.PNI	.SPA	.WIRE
.BDR	.DT	.HPG	.MNR	.PNM	.SPE	.WKS
.BEAN	.DT2	.HPGL	.MNT	.PNTG	.SPH	.WMA
.BIB	.DTA	.HPI	.MOBI	.PNZ	.SPJ	.WMDB
.BM2	.DTD	.HPL	.MOS	.POP	.SPP	.WMF
.BMP	.DTSX	.HS	.MOV	.POT	.SPQ	.WMV
.BMX	.DTW	.HTC	.MP3	.POTM	.SPR	.WN
.BNA	.DVI	.HTM	.MP4	.POTX	.SQB	.WP
.BND	.DVL	.HTML	.MPA	.PP4	.SQL	.WP4
.BOC	.DWG	.HWP	.MPF	.PP5	.SQLITE3	.WP5
.BOK	.DX	.HZ	.MPG	.PPAM	.SQLITEDB	.WP6
.BRD	.DXB	.I3D	.MPO	.PPM	.SR2	.WP7
.BRK	.DXF	.IB	.MRG	.PPS	.SRT	.WPA
.BRN	.DXL	.IBD	.MRXS	.PPSM	.SRW	.WPD
.BRT	.ECO	.IBOOKS	.MS11	.PPSX	.SSA	.WPE
.BSS	.ECW	.ICN	.MSG	.PPT	.SSK	.WPG
.BTD	.ECX	.ICON	.MSI	.PPTM	.ST	.WPL
.BTI	.EDB	.IDC	.MT9	.PPTX	.STC	.WPS
.BTR	.EFD	.IDEA	.MUD	.PRF	.STD	.WPT
.BZ2	.EGC	.IDX	.MWB	.PRIV	.STE	.WPW
.C	.EIO	.IFF	.MWP	.PRIVATE	.STI	.WRI
.C2	.EIP	.IGT	.MXL	.PRT	.STM	.WSC
.C4	.EIT	.IGX	.MYD	.PRW	.STN	.WSD
.C4D	.EMD	.IHX	.MYI	.PS	.STP	.WSF
.CAL	.EMF	.IIL	.MYL	.PSD	.STR	.WSH

.CALS	.EML	.IIQ	.NCR	.PSDX	.STW	.WTX
.CAN	.EMLX	.IMD	.NCT	.PSE	.STY	.WVL
.CD5	.EP	.INDD	.NDF	.PSID	.SUB	.X3D
.CDB	.EPF	.INFO	.NEF	.PSP	.SUMO	.X3F
.CDC	.EPP	.INK	.NFO	.PSPIMAG	.SVA	.XAR
.CDG	.EPS	.IPF	.NJX	E	.SVF	.XCODEPR
.CDMM	.EPSF	.IPX	.NLM	.PSW	.SVG	OJ
.CDMT	.EQL	.ITDB	.NOTE	.PTG	.SVGZ	.XDB
.CDR	.ERF	.ITW	.NOW	.PTH	.SWF	.XDL
.CDR3	.ERR	.IWI	.NRW	.PTX	.SXC	.XHTM
.CDR4	.ETF	.J2C	.NS2	.PU	.SXD	.XHTML
.CDR6	.ETX	.J2K	.NS3	.PVJ	.SXG	.XLC
.CDT	.EUC	.JAR	.NS4	.PVM	.SXI	.XLD
.CER	.EXR	.JAS	.NSF	.PVR	.SXM	.XLF
.CF	.FAL	.JAVA	.NV2	.PWA	.SXW	.XLGC
.CFG	.FAQ	.JB2	.NYF	.PWI	.T2B	.XLM
.CFM	.FAX	.JBMP	.NZB	.PWR	.TAB	.XLR
.CFU	.FB2	.JBR	.OBJ	.PXR	.TAR	.XLS
.CGI	.FB3	.JFIF	.OC3	.PY	.TBO	.XLSB
.CGM	.FBL	.JIA	.OC4	.PZ3	.TBK	.XLSM
.CIMG	.FBX	.JIS	.OC5	.PZA	.TBN	.XLSX
.CIN	.FCD	.JKS	.OCE	.PZP	.TCX	.XLT
.CIT	.FCF	.JNG	.OCI	.PZS	.TDF	.XLTM
.CKP	.FDB	.JOE	.OCR	.QCOW2	.TDT	.XLTX
.CLASS	.FDF	.JP1	.ODB	.QDL	.TE	.XLW
.CLKW	.FDR	.JP2	.ODG	.QMG	.TEX	.XML
.CMA	.FDS	.JPE	.ODM	.QPX	.TEXT	.XPM
.CMD	.FDT	.JPEG	.ODO	.QRY	.TF	.XPS
.CMX	.FDX	.JPG	.ODP	.QVD	.TFC	.XWP
.CNM	.FDXT	.JPG2	.ODS	.RA	.TG4	.XY3
.CNV	.FES	.JPS	.ODT	.RAD	.TGA	.XYP
.COLZ	.FFT	.JPX	.OFL	.RAR	.TGZ	.XYW
.CPC	.FH10	.JRTF	.OFT	.RAS	.THM	.YAL
.CPD	.FH11	.JS	.OMF	.RAW	.THP	.YBK
.CPG	.FH3	.JSP	.OPLC	.RCTD	.TIF	.YML
.CPP	.FH4	.JTX	.OQY	.RCU	.TIFF	.YSP
.CPS	.FH5	.JWL	.ORA	.RDB	.TJP	.YUV
.CPT	.FH6	.JXR	.ORF	.RDDS	.TLB	.Z3D
.CPX	.FH7	.KDB	.ORT	.RDL	.TLC	.ZABW
.CRD	.FH8	.KDBX	.ORX	.RFT	.TM	.ZDB
.CRT	.FIC	.KDC	.OTA	.RGB	.TM2	.ZDC
.CRWL	.FID	.KDI	.OTG	.RGF	.TMD	.ZIF
.CRYPT	.FIF	.KDK	.OTI	.RIB	.TMP	.ZIP
.CS	.FIG	.KES	.OTP	.RIC	.TMV	.ZIPX
.CSR	.FIL	.KEY		.RIFF		.ZW

3.2 EXTENSIÓN AÑADIDA A LOS ARCHIVOS CIFRADOS

3.2.1 PRIMERA Y SEGUNDA VERSIÓN

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

.crypt

3.2.2 TERCERA VERSIÓN

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

.cryptz

3.3 ARCHIVOS DE RESCATE

3.3.1 PRIMERA Y SEGUNDA VERSIÓN

El código dañino crea los siguientes archivos acerca del secuestro de los archivos del sistema comprometido.

[id_unico].html
[id_unico].bmp
!Recovery_[id_unico].bmp
!Recovery_[id_unico].html
!Recovery_[id_unico].txt

Los archivos se crean en las carpetas de Datos de Programas ("ProgramData"), en los directorios en los que se cifren archivos y en el escritorio del usuario. Un ejemplo del archivo HTML creado:

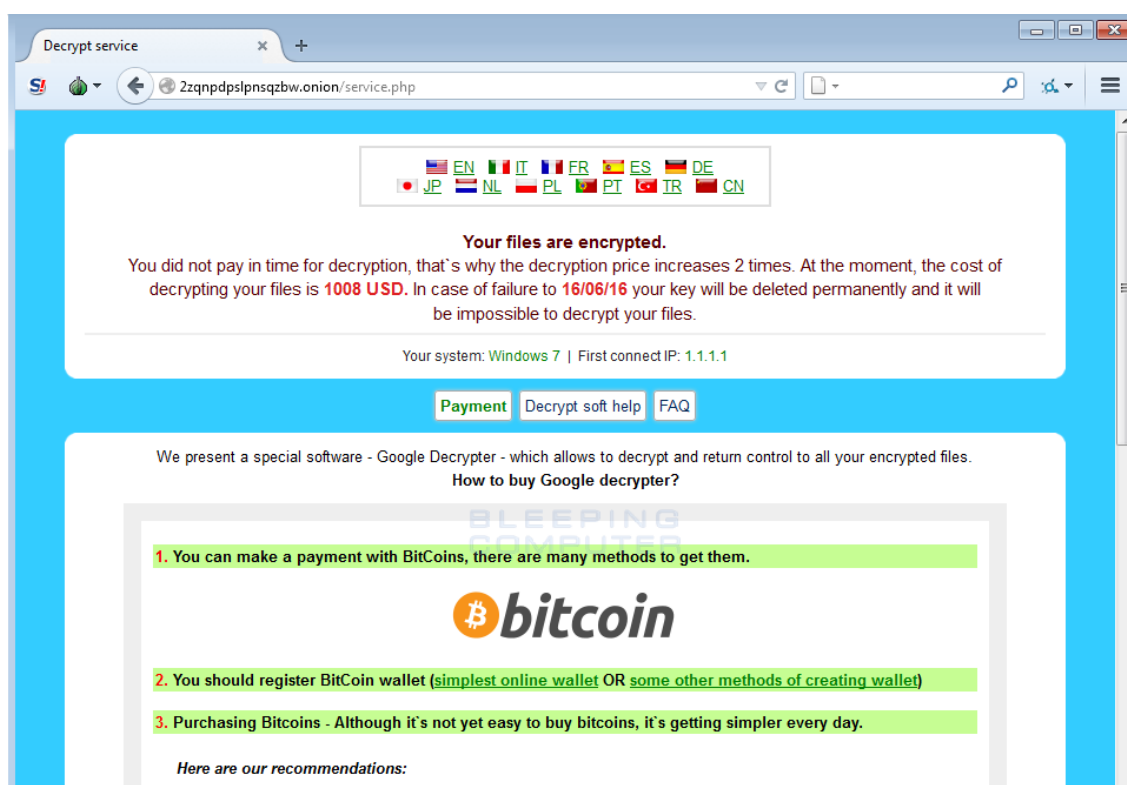


Ilustración 1. Archivo HTML con información del secuestro

Un ejemplo de fondo de pantalla del escritorio:

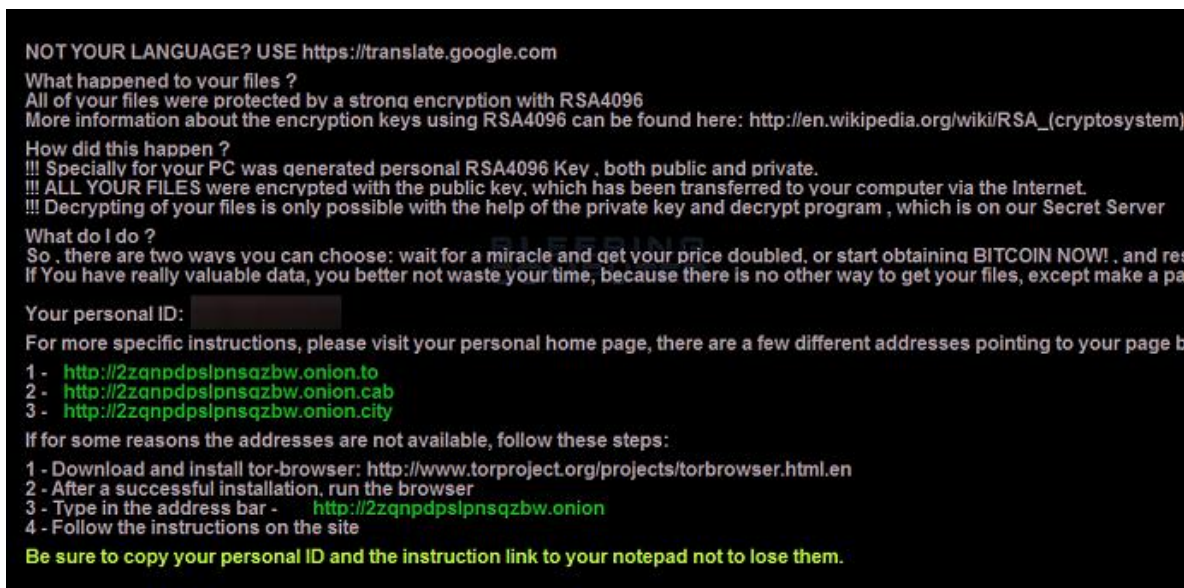


Ilustración 2. Ejemplo de fondo de escritorio con información del secuestro

3.3.2 TERCERA VERSIÓN

El código dañino crea los siguientes archivos acerca del secuestro de los archivos del sistema comprometido.

!.txt

!.bmp

!.html

!.bat

!.cfg

!H.lnk

!B.lnk

Los archivos con extensión ".bmp", ".html", ".cfg" y ".bat" son creados en la carpeta donde reside el código dañino. En el escritorio se crean todos los archivos menos el que tiene las extensiones ".bat" y ".cfg". En cada carpeta donde se cifren los archivos, se crean los archivos con extensión ".txt" y ".html". En dos carpetas especiales se crean los archivos con extensión ".bmp", ".html" y ".txt".

<raiz_del_sistema_operativo>:\Documents and Settings\<usuario>\NetHood

<raiz_del_sistema_operativo>:\Documents and Settings\<usuario>\Start Menu\Programs

A continuación se muestra un ejemplo de los archivos del secuestro.

@@@@@@ NOT YOUR LANGUAGE? USE <https://translate.google.com>

@@@@@@ What happened to your files ?

@@@@@@ All of your files were protected by a strong encryption with RZA4096

@@@@@@ More information about the en-Xryption keys using RZA4096 can be found here: [http://en.wikipedia.org/wiki/RSA_\(cryptosystem\)](http://en.wikipedia.org/wiki/RSA_(cryptosystem))

@@@@@@ How did this happen ?

@@@@@@ !!! Specially for your PC was generated personal RZA4096 Key , both publik and private.

@@@@@@ !!! ALL YOUR FILES were en-Xrypted with the publik key, which has been transferred to your computer via the Internet.

@@@@@@ !!! Decrypting of your files is only possible with the help of the privatt key and de-crypt program , which is on our Secret Server

@@@@@@ What do I do ?

@@@@@@ So , there are two ways you can choose: wait for a miracle and get your price doubled, or start obtaining BITCOIN NOW! , and restore your data easy way

@@@@@@ If You have really valuable data, you better not waste your time, because there is no other way to get your files, except make a payment

Your personal ID:

For more specific instructions, please visit your personal home page, there are a few different addresses pointing to your page below:

- 1 - <http://hn5fbbc4pyz77xfa.onion.to>
- 2 - <http://hn5fbbc4pyz77xfa.onion.cab>
- 3 - <http://hn5fbbc4pyz77xfa.onion.city>

If for some reasons the addresses are not available, follow these steps:

- 1 - Download and install tor-browser: <http://www.torproject.org/projects/torbrowser.html.en>
- 2 - After a successful installation, run the browser
- 3 - Type in the address bar - <http://hn5fbbc4pyz77xfa.onion>
- 4 - Follow the instructions on the site

Be sure to copy your personal ID and the instruction link to your notepad not to lose them.

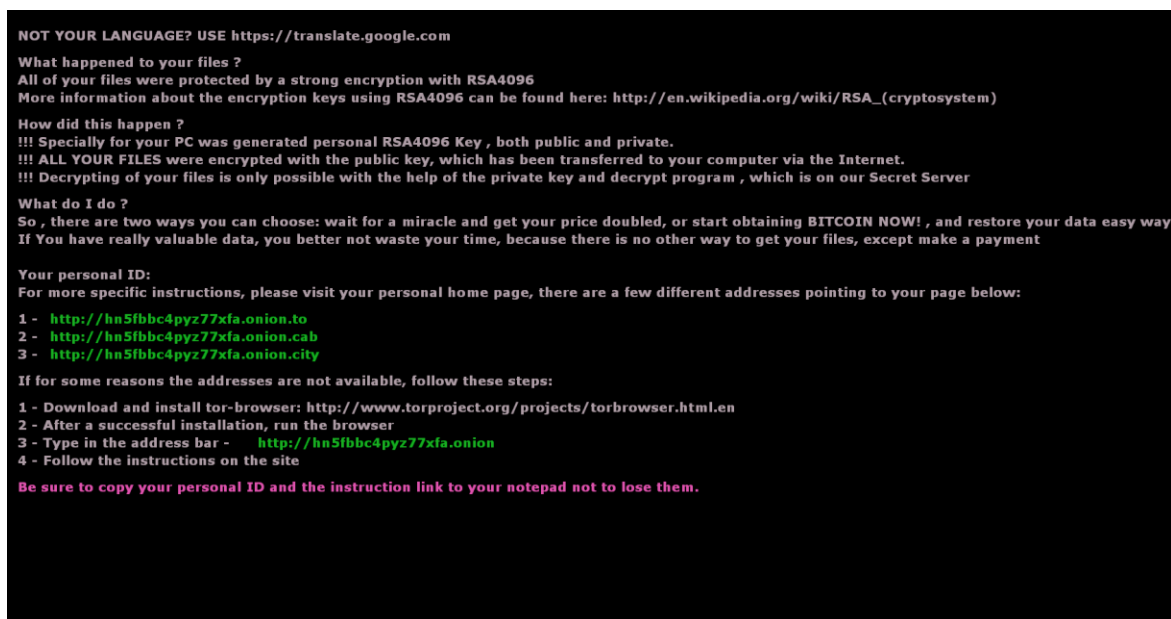


Ilustración 3. Ejemplo de fondo de escritorio con información del secuestro



Ilustración 4. Información acerca del secuestro de los archivos en formato HTML

En el caso de que se acceda mediante la red TOR a uno de los enlaces indicados se puede ver la siguiente interfaz:

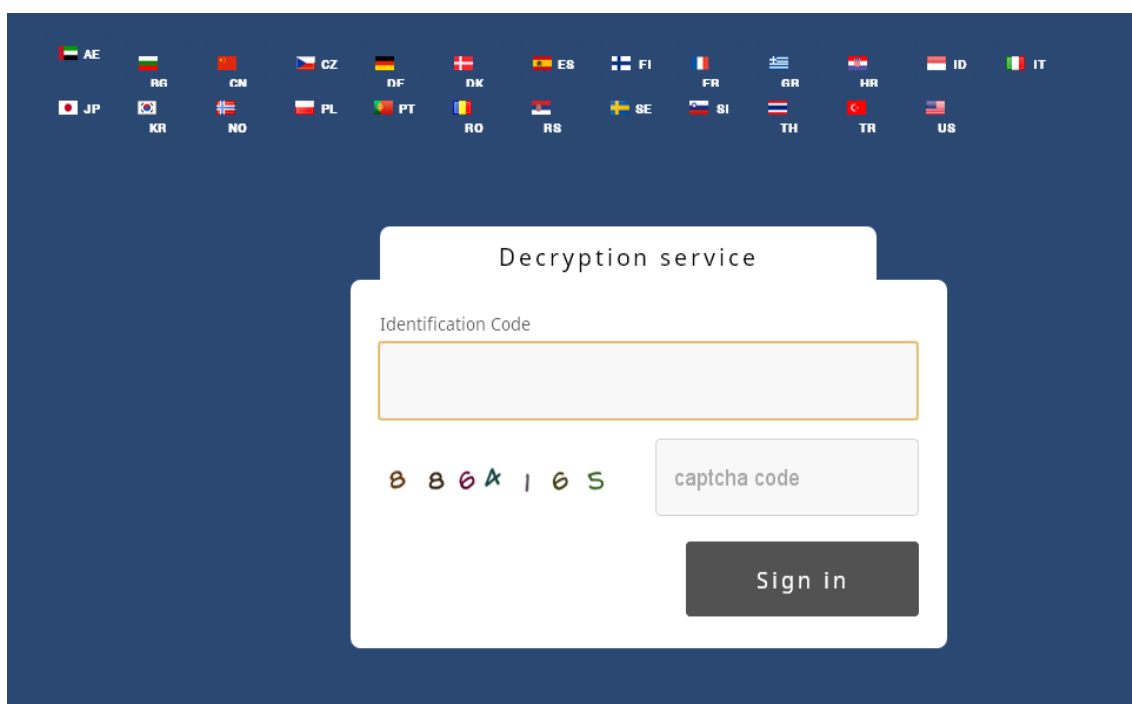


Ilustración 5. Página del servicio de descifrado en la red TOR

4. CARACTERÍSTICAS DEL CÓDIGO DAÑINO

El código dañino examinado posee las siguientes características:

- Carga el código dañino en el sistema.
- Enumera todas las unidades disponibles que sean de tipo disco duro, extraíble o recurso de red sobre las cuales cifrará todos los archivos que cumplan un patrón de extensión.
- Se comunica con un servidor C2.
- Posee 5 exportaciones con distintas funciones.
- Está programado en Borland Delphi, ya que se pueden apreciar restos de código de su librería estándar introducida por el compilador.
- Utiliza técnicas anti depuración.
- Enumera los procesos del sistema y termina algunos específicos.
- Crea archivos con información del secuestro de los archivos y los pasos a seguir para su recuperación.
- Modifica el fondo del escritorio con información acerca del secuestro de los archivos y el procedimiento a seguir para recuperarlos.

5. DETALLES GENERALES

La muestra analizada se corresponde con la siguiente firma MD5:

d01fd2bb8c6296d51be297978af8b3a1 → "dropper" original
 ae06248ab3c02e1c2ca9d53b9a155199 → código dañino extraído del "dropper"

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 15 de junio del 2016.

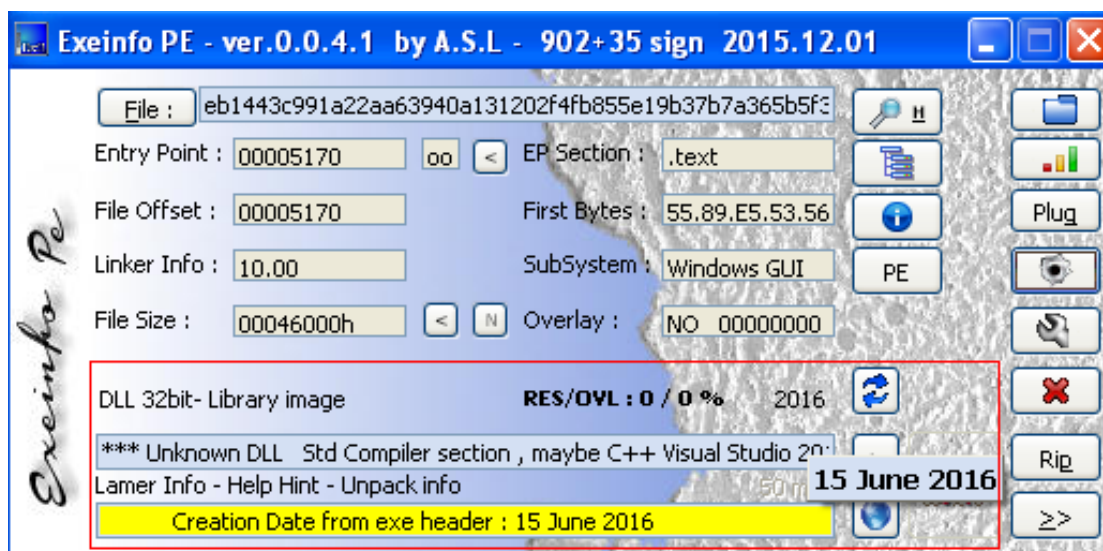


Ilustración 6. Detalles del binario

6. PROCEDIMIENTO DE INFECCIÓN

La infección en el equipo se produce al ejecutar el fichero que contiene el código dañino en sí mismo, por ejemplo:

- Visitando una página web comprometida con un "Exploit Kit", con un navegador vulnerable que permita la ejecución del código dañino.
- Ejecutando un programa de tipo descargador ("downloader") que descargue el binario de Internet y lo ejecute.
- Ejecutando el código dañino directamente, si se realizó una descarga por una red P2P de algún binario que lo lleve embebido.

Una vez ejecutado el código dañino realiza las siguientes acciones en el equipo de la víctima:

- El código dañino es un "dropper" que descifra en memoria el código dañino final, el "ransomware", y lo ejecuta.

- El "dropper" inyecta el código dentro del primer proceso del sistema que coincida con los siguientes:

explorer.exe	regsvr32.exe
--------------	--------------

- El código dañino se asegura que se esté ejecutando en alguno de los procesos anteriormente indicados y, en caso contrario, finaliza su ejecución. Esto se utiliza como técnica anti-depuración ya que, por ejemplo, *OllyDbg* utiliza el ejecutable "LOADDLL.EXE" como cargador de una librería bajo el contexto de su proceso.
- El código dañino tiene 5 exportaciones que serán utilizadas para afectar con su carga dañina el sistema. Sus nombres son:

XXS0S
XXS1S
XXS2S
XXS3S
XXS4S

- El código dañino crea una copia del archivo "rundll32.exe" y utiliza dicha copia para ejecutarse invocando una determinada función exportada cada vez. La copia del archivo se llama "explorer.exe".

7. CARACTERÍSTICAS TÉCNICAS

El código dañino se encuentra embebido en un "dropper" propietario de alto nivel de sofisticación que es una librería dinámica ("DLL").

Una vez quitada esa primera capa inicial, el código dañino procede a comprobar que se esté ejecutando desde un proceso llamado "explorer.exe" o "regsvr32.exe", comprobando tanto con los nombres en minúsculas como en mayúsculas. Los procesos pertenecen al Explorador de Windows y al registrador de librerías dinámicas ("DLL") del sistema operativo respectivamente.

Para realizar esta comprobación el código dañino utiliza la función "GetModuleFileNameA" que devuelve en un *buffer* destino la ruta y nombre del ejecutable que la llama. En el caso de que no se encuentre en alguno de ellos, el código dañino procede a finalizar su ejecución. Esta comprobación indica que el "dropper" necesita de un vector de entrada para realizar correctamente el proceso de infección, es decir, hace falta un ejecutable anterior que ponga en marcha todo el proceso.

A continuación, se chequea que el sistema operativo sea un sistema operativo de 64 bits mediante la función "IsWow64Process". Dependiendo del resultado de esta llamada prepara una ruta hacia el archivo "rundll32.exe" accediendo al directorio de

Windows mediante la función "GetWindowsDirectory" y concatenando ambas cadenas de texto.

Crea una copia del archivo "rundll32.exe" en la misma ubicación del código dañino bajo el nombre "explorer.exe" y procede a crear una cadena de llamada de la siguiente forma:

<ruta_hacia_el_rundll32_recien_creado>\<el código dañino>,<función exportada>

El código dañino exporta 5 funciones:

XXS0S

XXS1S

XXS2S

XXS3S

XXS4S

La primera de las exportaciones invocada es la "XXS0S". La función principal del ejecutable "rundll32.exe" recién creado es permitir ejecutar una función exportada de la librería y de esta forma el código dañino comienza todo su proceso real. Tras realizar esta llamada el código dañino finaliza su ejecución.

7.1 EXPORTACIÓN XXS0S

La primera acción de la exportación es comprobar si un *flag* que hay en memoria está a "1", que por defecto está a "0". En el caso de que sea "0" procederá a relanzar el código dañino mediante la copia de "rundll32.exe", tal y como hizo antes, pero a la exportación "XXS1S". En ningún momento del análisis del presente informe se vio modificado ese *flag* desde el valor inicial a "1".

Sin embargo, si el *flag* está a "1" el código dañino procede a crear dos hilos. El primero de ellos tiene la función de comenzar a enumerar todas las ventanas existentes en el sistema y comprobar cada una de ellas si son visibles o no. En el caso de que no lo sean, procede a analizar la siguiente ventana; y en el caso de que sea visible, se obtiene el identificador del proceso al que pertenece la ventana mediante la función "GetWindowThreadProcessId".

A continuación, obtiene una lista de las funciones que le harán falta mediante "GetProcAddress" de la librería "kernel32.dll":

CreateToolhelp32Snapshot	Process32First	Thread32Next
Heap32ListFirst	Process32Next	Module32First
Heap32ListNext	Process32FirstW	Module32Next
Heap32First	Process32NextW	Module32FirstW
Heap32Next	Thread32First	Module32NextW
Toolhelp32ReadProcessMemory		

Una vez obtenidas procede a crear una imagen de los procesos del sistema mediante la función "CreateToolhelp32Snapshot", "Process32First" y "Process32Next".

Por cada uno de los procesos encontrados se comprueba su identificador de proceso (PID), con el del proceso al que pertenece la ventana encontrada anteriormente.

En el caso de que el identificador del proceso coincida, comprueba que el nombre del proceso sea "consent.exe". En el caso de que no lo sea, pasará a obtener el siguiente proceso. Cuando coincida, procederá a enviar el siguiente comando del sistema al proceso:

```

00030044 | hWnd = 30044
00000112 | Message = WM_SYSCOMMAND
0000F120 | Type = SC_RESTORE
00030044 | X = 68. Y = 3.
  
```

Ilustración 7. Envío de comando al proceso

En el análisis nunca se dio el caso de que coincidiese el proceso con el buscado con lo que no se pudo saber qué esperaba obtener con dicho comando. El proceso "consent.exe" existe a partir de Windows Vista y es el encargado de visualizar la ventana de autorización para permitir que una aplicación pueda efectuar cambios en el sistema si la UAC está habilitada.

El hilo repite la búsqueda de ventanas con un descanso de 1 segundo entre iteración e iteración mediante "Sleep".

El segundo hilo de la exportación utiliza el comando "runas" para ejecutar la copia de "rundll32.exe" renombrada a "explorer.exe" en conjunto con la ruta al código dañino y la exportación "XXS1S". De esta forma ejecuta la exportación con privilegios más elevados. La ejecución del comando la realiza mediante "ShellExecuteEx".

7.2 EXPORTACIÓN XXS1S

Según se ejecuta la exportación se descifra el siguiente texto:

FX1112

Con la cadena de texto descifrada, el código dañino crea un "Mutex" con ese nombre:

```

00000000 | pSecurity = NULL
00000000 | InitialOwner = FALSE
0094000C | MutexName = "FX1112"
  
```

Ilustración 8. Creación del Mutex FX1112

Posteriormente, comprueba el último error del sistema mediante "GetLastError" y en el caso de que obtenga "ERROR_ALREADY_EXISTS" (0xb7) o "ACCESS_DENIED" (0x05), el código dañino finaliza su ejecución.

Tras ello crea un identificador único de la víctima mediante un algoritmo propio usando como semilla aleatoria "GetTickCount".

A continuación, descifra una serie de cadenas de texto desde unas tablas embebidas en su código de cadenas cifradas.

\WINDOWS\	\PERFLOGS\
\WINNT	\EFI\
\RECYCLER\	\CONFIG.MSI\
\SYSTEM~1\	\PROGRA~1\
\BOOT\	\PROGRA~2\
\RECOVERY\	\GOOGLE\
\\$RECYCLE.BIN\	\TEMP\

La segunda tabla también tiene rutas y algún archivo. Esta tabla es usada posteriormente como lista negra: cualquier archivo que se encuentre en alguna de esas rutas, o tenga ese nombre de archivo, será ignorado en el proceso de cifrado.

\F4BC~1\	AUTOEXEC.BAT
\ALLUSE~1\	THUMBS.DB
\PROGRA~1\	\APPLIC~1\
\PROGRA~2\	\COOKIES\
\APPDATA\	\LOCALS~1\
\PROGRA~3\	\TEMPLA~1\
\PUBLIC\	

La tercera tabla que descifra son las extensiones de los archivos que cifrará que suman un total de 934 y se pueden revisar en el apartado [3.1 EXTENSIONES A CIFRAR](#).

Tras descifrar las tablas procede a relanzarse de nuevo usando el "rundll32.exe" renombrado a "explorer.exe", pero esta vez a la exportación "XXS4S" mediante "CreateProcessW". Tras la creación de ese hilo se vuelve a relanzar de nuevo usando el "rundll32.exe" renombrado a "explorer.exe" con la exportación objetivo "XXS2S" mediante "CreateProcessW".

Posteriormente, descifra una cadena de texto cuyo resultado es un certificado en base64.

El código dañino busca en su mismo directorio un archivo llamado "!.cfg" de forma que si se encuentra será leído y en caso contrario procederá a crearlo. Como contenido, el archivo lleva el certificado en base64 pero cifrado mediante una simple operación XOR con el valor "0xE".

A continuación, descifra en memoria todas las cadenas de texto sobre el secuestro de los archivos, enlaces a las webs ".onion" y los pasos a seguir para poder recuperarlos. Una vez descifradas estas cadenas, crea en memoria un archivo HTML y

un archivo BMP con dicha información. Con esta información crea dos archivos en el directorio donde se encuentre, uno llamado "!.html" y otro "!.bmp". También crea copias de estos archivos en las siguientes rutas especiales:

<raiz_del_sistema_operativo>:\Documents and Settings\<usuario>\NetHood
<raiz_del_sistema_operativo>:\Documents and Settings\<usuario>\Start Menu\Programs

En dichas rutas especiales también crea un archivo de texto llamado "!.txt" con la información del secuestro de los archivos.

Tras crear los archivos el código dañino procede a obtener la clave pública. Para ello adquiere el contexto mediante "CryptAcquireContextA" y obtiene la clave RSA desde el certificado en base64 mediante "CryptStringToBinaryA". Esta clave es usada posteriormente con la función "CryptImportKey".

A continuación, procede a obtener el tipo de todas las unidades comenzando desde la "Z" hasta la "A" en un bucle. Por cada una de las unidades comprobadas solo busca aquellas que sean de tipo disco duro. La comprobación la realiza mediante "GetDriveType".

```

call     00384850
mov     eax, [ebp-8]
call     003849FC
push    eax
call     <jmp.&kernel32.GetDriveTypeA>
009468AC  LRootPathName = "H:\\"
0006F414  Pointer to next SEH record
003B273F  SE handler
  
```

Ilustración 9. Obteniendo el tipo de todas las posibles unidades en el sistema

Tras esta acción, procede a obtener todos los archivos y carpetas de las unidades de discos duros mediante "FindFirstFile" y "FindNextFile". Por cada uno de los archivos encontrados se procede a comprobar que no se encuentre en ninguna de las rutas de la tabla de rutas prohibidas o que sea alguno de los archivos indicado en esa tabla.

00940168	ASCII "\\F4BC~1\\"
00940180	ASCII "\\ALLUSE~1\\"
00940198	ASCII "\\PROGRA~1\\"
009401B0	ASCII "\\PROGRA~2\\"
009401C8	ASCII "\\APPDATA\\"
009401E0	ASCII "\\PROGRA~3\\"
009401F8	ASCII "\\PUBLIC\\"
00940210	ASCII "AUTOEXEC.BAT"
0094022C	ASCII "THUMBS.DB"
00940244	ASCII "\\APPLIC~1\\"
0094025C	ASCII "\\COOKIES\\"
00940274	ASCII "\\LOCALS~1\\"
0094028C	ASCII "\\TEMPLA~1\\"

Ilustración 10. Lista de rutas y archivos no permitidos

En el caso de que no sea así, se comprueba que su extensión no contenga la cadena ".cryp" para excluirlo del proceso de cifrado. Después, comprueba la extensión del archivo con la lista de extensiones a afectar, [3.1 EXTENSIONES A CIFRAR](#). En el caso de que no coincida con alguna de ellas el archivo también será ignorado.

Por cada uno de los ficheros obtiene los atributos del archivo mediante "GetFileAttributesExW" y los asigna a "ARCHIVO" mediante "SetFileAttributesExW".

```

mov     ebx, eax
push    ebx
call    <jmp.&kernel32.GetFileAttributesExW>
push    20      -> ARCHIVO
push    ebx
call    <jmp.&kernel32.SetFileAttributesW>
push    0
push    80
push    3
push    0
push    3
push    C0000000
mov     eax, [ebp-4]
call    00384DD8
push    eax
call    <jmp.&kernel32.CreateFileW>
  
```

Ilustración 11. Cambio de los atributos del archivo a cifrar

Una vez cambiados los atributos se genera una clave aleatoria mediante un algoritmo propietario usando como semillas "GetTickCount". Con ella se procede a cifrar el archivo usando el algoritmo RC4 y un algoritmo propietario de operación XOR. Tras cifrar todo el archivo, se le añade al final del archivo una cabecera con información acerca del archivo como su tamaño original, la clave RC4 utilizada cifrada mediante RSA.

Tras ello se le añade al archivo la extensión:

.cryptz

Una vez realizado, se procede a continuar con el siguiente archivo hasta finalizar toda la unidad. Cuando acaba con todas las unidades de tipo disco duro procede igualmente con todas las unidades disponibles de tipo extraíble y de recursos de red.

Posteriormente, enumera todos los recursos de red que puedan no estar mapeados a una unidad lógica pero sí accesibles y por cada uno de ellos que encuentre procederá de igual manera que con las unidades anteriormente citadas. Esta enumeración la realiza mediante las funciones de la librería "mpr.dll":

WNetOpenEnumA
WNetCloseEnum
WNetEnumResourceA

Una vez el código dañino finaliza, modifica el fondo del escritorio con el archivo ".bmp" con la información acerca del secuestro de los archivos y el procedimiento a seguir para recuperarlos.

Por último, crea en el escritorio un archivo ".bmp", ".txt", un enlace directo al archivo ".bmp", un archivo ".html" y otro enlace directo al ".html" donde todos ellos hacen referencia al secuestro de los archivos. También se crea un archivo de procesamiento por lotes llamado "!.bat".

La función del archivo de procesamiento por lotes es borrar el código dañino del disco y posteriormente eliminarse a sí mismo pero, por la exportación "XXS4S" que continuamente cierra determinados procesos entre los que está incluido el Terminal de Windows, según se arranca el archivo, el proceso lo finaliza no pudiendo borrar el código dañino.

```

:St
Erase <ruta_al_código_dañado>
If exist <ruta_al_código_dañado> Goto St
Erase "!..bat"
  
```

7.3 EXPORTACIÓN XXS2S

El código dañino inicializa Winsock 2.0 según entra en la función exportada. Una vez inicializado procede a descifrar unas cadenas de texto entre las que destaca el nombre de una librería dinámica ("DLL"):

```

stillerzzz.dll
  
```

Tras ello crea una conexión con la dirección IP 188.0.236.7 y solicita que le envíe información. Si la dirección IP no responde, se queda a la espera en un bucle infinito. Lo que espera esta petición es la librería para ejecutarla en el sistema.

7.4 EXPORTACIÓN XXS3S

Esta exportación tiene como única función poner un *flag* a "1" e invocar la exportación "XXS1S". Ese *flag* impide que la exportación "XXS1S" llame a la exportación "XXS2S". Tras ello finaliza.

7.5 EXPORTACIÓN XXS4S

En esta exportación el código dañino realiza de forma continua una enumeración de todos los procesos del sistema y, en caso de que encuentre alguno de la siguiente lista, procederá a terminarlo de forma instantánea mediante "TerminateProcess". Esta comprobación la realiza cada 200 milisegundos.

```

WerFault.exe
taskmgr.exe
msconfig.exe
cmd.exe
  
```

Los procesos indicados en la tabla corresponden a "Microsoft Error Reporting", al Gestor de Tareas, el Editor de Arranque y Sistema, y el Terminal de Windows respectivamente.

8. CIFRADO Y OFUSCACIÓN

8.1 CIFRADO

El código dañino cifra los archivos mediante tres algoritmos: RC4, RSA y un algoritmo propietario XOR.

El código dañino comprueba el tamaño del archivo a cifrar antes de comenzar el proceso y, si es más grande de 13 MB, establece que su tamaño son 13 MB. También cambia los atributos de los archivos a ser cifrados, primero a modo "archivo", y posteriormente a "solo lectura".

Por cada archivo a cifrar se genera una clave aleatoria mediante algoritmo propietario que utiliza como semilla llamadas a la función "GetTickCount". Con esta clave se cifra el contenido del archivo usando el algoritmo RC4 tras el algoritmo propietario XOR.

Posteriormente, se cifra la clave RC4 con la clave pública RSA que lleva embebida el código dañino. Esta clave es una secuencia de bytes en base64, la cual es convertida e importada mediante las funciones criptográficas de Microsoft.

```

push    eax
push    0
push    -1
push    0
mov     eax, [ebp-24]
push    eax
call    <jmp.&advapi32.CryptEncrypt>
test    eax, eax
00000000
009697A8 ASCII "PSHKL[?D0JWZm3<QN$gE%wZZU@GqNLt~FLD)s:U,B10+0p$0+?6NG0Jk*$tEq&Xh

```

Ilustración 12. Cifrado de la clave AES usando RSA

Una vez se ha cifrado la clave RC4, se añade al final del archivo cifrado una cabecera con dicha clave e información del archivo original como su tamaño. Por último, se le añade la extensión indicada en el apartado [3.2 EXTENSIÓN AÑADIDA](#) del presente informe mediante la función "MoveFileW".

```

ExistingName = "C:\1.js"
NewName     = "C:\1.js.cryp2"

```

Ilustración 13. Añadido de la extensión de cifrado

8.2 OFUSCACIÓN

El código dañino tiene ofuscados una gran cantidad de cadenas de texto para evitar el análisis estático o el visionado con un editor hexadecimal. El algoritmo usado es privativo siendo una simple operación XOR. El prototipo de la función es el siguiente:

```
void desofuscarCadena(char key, char* cadena_original, char* cadena_final)
```

El algoritmo es simple pero efectivo:

```

if ( v5 > 0 )
{
    v6 = 1;
    do
    {
        *(_BYTE *)( + v6 - 1 ) = v3 ^ *(_BYTE *)(v10 + v6 - 1);
        ++v6;
        --v5;
    }
    while ( v5 );
}

```

9. PERSISTENCIA EN EL SISTEMA

El código dañino no realiza ninguna acción para tener persistencia en el sistema.

10. CONEXIONES DE RED

El código dañino establece conexión con su dominio C2, intentándolo de forma periódica en caso de no conectar.

El código dañino utiliza *sockets* creando una conexión completa con el servidor remoto tal y como se puede apreciar en la siguiente captura aunque el servidor cierra la conexión de forma automática.

Filter: tcp.stream eq 3					
Expression... Clear Apply Save					
No.	Time	Source	Destination	Protocol	Length Info
1362	250.023638	192.168.1.102	188.0.236.7	TCP	62 1137-443 [SYN] Seq=0 win=64240 Len=0 MSS=1460 SACK_PERM=1
1363	250.023890	188.0.236.7	192.168.1.102	TCP	62 443-1137 [SYN, ACK] Seq=0 Ack=1 win=14600 Len=0 MSS=1460 SACK_PERM=1
1364	250.023915	192.168.1.102	188.0.236.7	TCP	54 1137-443 [ACK] Seq=1 Ack=1 win=64240 Len=0
1366	250.026065	192.168.1.102	188.0.236.7	SSL	106 Continuation Data
1367	250.026275	188.0.236.7	192.168.1.102	TCP	60 443-1137 [ACK] Seq=1 Ack=53 win=14600 Len=0
1392	250.943061	192.168.1.102	188.0.236.7	SSL	58 Continuation Data
1393	250.943462	188.0.236.7	192.168.1.102	TCP	60 443-1137 [ACK] Seq=1 Ack=57 win=14600 Len=0
1394	250.943516	192.168.1.102	188.0.236.7	SSL	313 Continuation Data
1395	250.943728	188.0.236.7	192.168.1.102	TCP	60 443-1137 [ACK] Seq=1 Ack=316 win=15544 Len=0
1523	261.026153	188.0.236.7	192.168.1.102	TCP	60 443-1137 [FIN, ACK] Seq=1 Ack=316 win=15544 Len=0
1524	261.026178	192.168.1.102	188.0.236.7	TCP	54 1137-443 [ACK] Seq=316 Ack=2 win=64240 Len=0
1526	261.027667	192.168.1.102	188.0.236.7	TCP	54 1137-443 [FIN, ACK] Seq=316 Ack=2 win=64240 Len=0
1527	261.027870	188.0.236.7	192.168.1.102	TCP	60 443-1137 [ACK] Seq=2 Ack=317 win=15544 Len=0

Ilustración 14. Comunicación con el servidor C2 sin éxito

Tal y como se puede apreciar en la ilustración superior, la única comunicación que llega a establecerse con el dominio C2 es un "handshake" y de forma automática el servidor cierra la conexión.

La respuesta que espera el código dañino es una librería, "stillerzz.dll", que, en caso de ser recibida, se pondría en ejecución.

11. ARCHIVOS RELACIONADOS

<%DESKTOP%>			
Nombre	Fecha Creación	Tamaño bytes	Hash SHA1
!.bmp	<varía>	<varía>	<varía>
!.html	<varía>	<varía>	<varía>
!.txt	<varía>	<varía>	<varía>
!H.lnk	<varía>	<varía>	<varía>
!.B.lnk	<varía>	<varía>	<varía>
!Recovery_[id_unico].bmp (v1 y 2)	<varía>	<varía>	<varía>
!Recovery_[id_unico].html (v1 y 2)	<varía>	<varía>	<varía>
!Recovery_[id_unico].txt (v1 y 2)	<varía>	<varía>	<varía>
<ruta_del_código_dañino>			
Nombre	Fecha Creación	Tamaño bytes	Hash SHA1
!.bat	<varía>	<varía>	<varía>
!.cfg	<varía>	<varía>	<varía>
explorer.exe	<varía>	<varía>	<varía>
<código_dañino>	<varía>	286720 bytes	6be21e389056ca028cf9083e42a765e8f61b0b5c
<varía>			
Nombre	Fecha Creación	Tamaño bytes	Hash SHA1
!.html	<varía>	<varía>	<varía>
!.txt	<varía>	<varía>	<varía>
!Recovery_[id_unico].bmp (v1 y 2)	<varía>	<varía>	<varía>
!Recovery_[id_unico].html (v1 y 2)	<varía>	<varía>	<varía>
!Recovery_[id_unico].txt (v1 y 2)	<varía>	<varía>	<varía>
<%STARTUP%\Programas>			
Nombre	Fecha Creación	Tamaño bytes	Hash SHA1
!.bmp	<varía>	<varía>	<varía>
!.html	<varía>	<varía>	<varía>
!.txt	<varía>	<varía>	<varía>
<%NETHOOD%>			
Nombre	Fecha Creación	Tamaño bytes	Hash SHA1
!.bmp	<varía>	<varía>	<varía>
!.html	<varía>	<varía>	<varía>
!.txt	<varía>	<varía>	<varía>
<%PROGRAMDATA%>			
Nombre	Fecha Creación	Tamaño bytes	Hash SHA1
[id_unico].html	<varía>	<varía>	<varía>
[id_unico].bmp	<varía>	<varía>	<varía>

12. DETECCIÓN

Para detectar si un equipo se encuentra, o ha estado infectado, bien 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. También se puede recurrir a observar ciertos archivos del sistema.

12.1 PRIMERA Y SEGUNDA VERSIÓN

La existencia de los siguientes archivos en el sistema indica compromiso por parte del código dañino.

```
[id_unico].html
[id_unico].bmp
!Recovery_[id_unico].bmp
!Recovery_[id_unico].html
!Recovery_[id_unico].txt
```

El cambio de fondo del escritorio a una imagen con información del secuestro también es un claro compromiso del sistema.

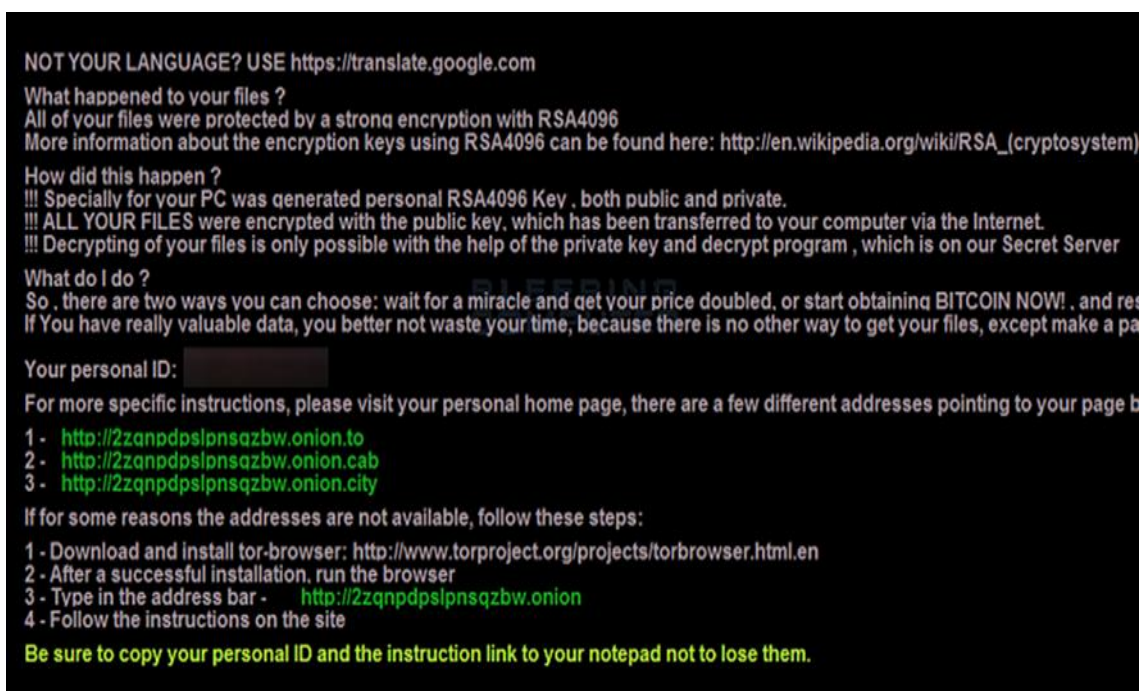


Ilustración 15. Fondo del escritorio cambiado a información del secuestro

La presencia de archivos con extensión doble, siendo la segunda extensión ".crypt" implica un compromiso del sistema, especialmente si dichos archivos ya no pueden abrirse con las aplicaciones con las que usualmente se podían abrir.

12.2 TERCERA VERSIÓN

La existencia de los siguientes archivos puede indicar el compromiso del sistema por parte del código dañino.

```
!.txt
!.bmp
!.html
!.bat
!.cfg
!H.lnk
!B.lnk
```

Si en las siguientes carpetas se encuentra alguno de los archivos previamente indicados en la tabla, es un claro indicio de compromiso del sistema:

```
<raiz_del_sistema_operativo>:\Documents and Settings\<usuario>\NetHood
<raiz_del_sistema_operativo>:\Documents and Settings\<usuario>\Start Menu\Programs
```

El cambio de fondo del escritorio a la imagen siguiente también es un claro compromiso del sistema.

```
NOT YOUR LANGUAGE? USE https://translate.google.com

What happened to your files ?
All of your files were protected by a strong encryption with RSA4096
More information about the encryption keys using RSA4096 can be found here: http://en.wikipedia.org/wiki/RSA_(cryptosystem)

How did this happen ?
!!! Specially for your PC was generated personal RSA4096 Key , both public and private.
!!! ALL YOUR FILES were encrypted with the public key, which has been transferred to your computer via the Internet.
!!! Decrypting of your files is only possible with the help of the private key and decrypt program , which is on our Secret Server

What do I do ?
So , there are two ways you can choose: wait for a miracle and get your price doubled, or start obtaining BITCOIN NOW! , and restore your data easy way
If You have really valuable data, you better not waste your time, because there is no other way to get your files, except make a payment

Your personal ID:
For more specific instructions, please visit your personal home page, there are a few different addresses pointing to your page below:
1 - http://hn5fbbc4pyz77xfa.onion.to
2 - http://hn5fbbc4pyz77xfa.onion.cab
3 - http://hn5fbbc4pyz77xfa.onion.city

If for some reasons the addresses are not available, follow these steps:
1 - Download and install tor-browser: http://www.torproject.org/projects/torbrowser.html.en
2 - After a successful installation, run the browser
3 - Type in the address bar - http://hn5fbbc4pyz77xfa.onion
4 - Follow the instructions on the site

Be sure to copy your personal ID and the instruction link to your notepad not to lose them.
```

Ilustración 16. Fondo del escritorio cambiado a información del secuestro

La presencia de archivos con extensión doble, siendo la segunda extensión ".cryptz" implica un compromiso del sistema, especialmente si dichos archivos ya no pueden abrirse con las aplicaciones con las que usualmente se podían abrir.

Se puede comprobar el compromiso del sistema, siempre que el código dañino esté en ejecución, al lanzar cualquiera de las siguientes aplicaciones y observando si finalizan solas a los pocos instantes:

WerFault.exe
taskmgr.exe
msconfig.exe
cmd.exe

12.3 MANDIANT

Se ha generado un nuevo archivo indicador de compromiso. El nombre del indicador generado es con GUID "cf422fca-efcc-4038-8dc9-85fef0241b13".

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

13.1 PRIMERA Y SEGUNDA VERSIÓN

Para desinfectar el sistema del código dañino se procederá a buscar en todo el equipo y borrar los archivos de la siguiente tabla:

[id_unico].html
[id_unico].bmp
!Recovery_[id_unico].bmp
!Recovery_[id_unico].html
!Recovery_[id_unico].txt

Para descifrar los archivos se puede utilizar la herramienta creada por Kaspersky "Rannoh Decryptor".

13.1.1 KASPERSKY RANNOH DECRYPTOR

La herramienta se puede descargar de la web de Kaspersky¹ y uso es simple: se debe pulsar el botón de "Start Scan" y la aplicación solicitará que se le aporte un archivo cifrado a elección del usuario y posteriormente el mismo archivo descifrado. La

¹ <http://support.kaspersky.com/mx/8547>

herramienta funciona mejor cuanto más grande sea la muestra aportada y del mismo modo se requiere una copia sin cifrar de dicho archivo.

Una vez aportados ambos archivos buscará todos los archivos con extensión “.crypt” e intentará descifrarlos. Por defecto no borra los archivos cifrados para evitar pérdida de datos por errores de descifrado.

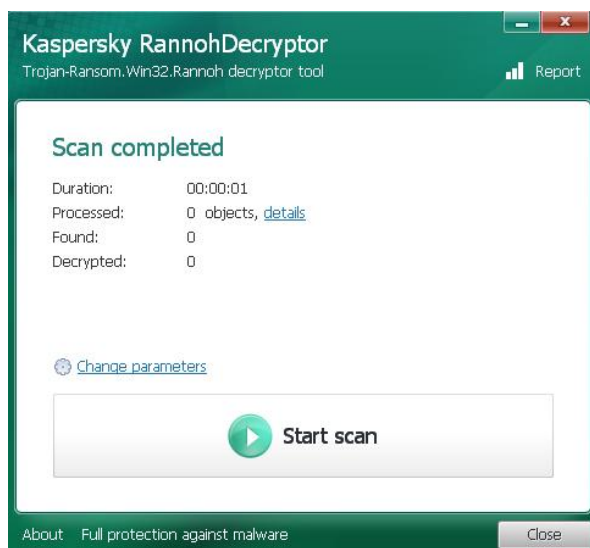


Ilustración 17. Herramienta gratuita de descifrado de primera y segunda versión

13.2 TERCERA VERSIÓN

Para desinfectar el sistema del código dañino se procederá a buscar en todo el equipo y borrar los archivos de la siguiente tabla:

!.txt
!.bmp
!.html
!.bat
!.cfg
!H.lnk
!B.lnk

También se debería cambiar el fondo de escritorio a un fondo adecuado al usuario.

El código dañino no borra las Shadow Copy por lo que, si están activas, se pueden recuperar los archivos gracias a ellas con herramientas “Shadow Explorer”.

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. INFORMACIÓN DEL ATACANTE

14.1 188.0.236.7

La información WHOIS de la dirección IP 188.0.236.7 es la siguiente:

IP Location	 Moldova, Republic Of Chisinau Sicres S.r.l.
ASN	 AS203912 SICRES-AS , MD (registered Oct 06, 2015)
Whois Server	whois.ripe.net
IP Address	188.0.236.7


```
% Abuse contact for '188.0.224.0 - 188.0.239.255' is ' admin@sicres.md '

inetnum:        188.0.224.0 - 188.0.239.255
netname:        SICRES
country:        MD
org:            ORG-SICR1-RIPE
admin-c:        AD12788-RIPE
tech-c:        SSID2-RIPE
status:        ASSIGNED PI
mnt-by:        RIPE-NCC-END-MNT
mnt-by:        SICRES-MNT
notify:        admin@sicres.md
created:        2015-10-05T10:53:06Z
last-modified:  2016-04-14T10:47:39Z
source:        RIPE
sponsoring-org: ORG-AC2-RIPE

organisation:   ORG-SICR1-RIPE
org-name:       "SICRES" S.R.L.
org-type:       OTHER
address:        MD-2004, Dosoftei 122, Chisinau, Republic of Moldova
e-mail:         admin@sicres.md
abuse-c:        AR33644-RIPE
mnt-ref:        SICRES-MNT
mnt-by:        SICRES-MNT
created:        2015-09-29T06:44:14Z
last-modified:  2015-09-29T07:09:08Z
source:        RIPE

person:         Alexander Dragan
address:        MD-2004, Dosoftei 122, Chisinau, Republic of Moldova
phone:          +37322885885
nic-hdl:        AD12788-RIPE
mnt-by:        SICRES-MNT
e-mail:         dragan@sicres.md
created:        2015-09-29T12:25:59Z
last-modified:  2015-09-29T12:25:59Z
```

Ilustración 18. Información WHOIS de la IP 188.0.236.7

14.1.1 GEOLOCALIZACIÓN

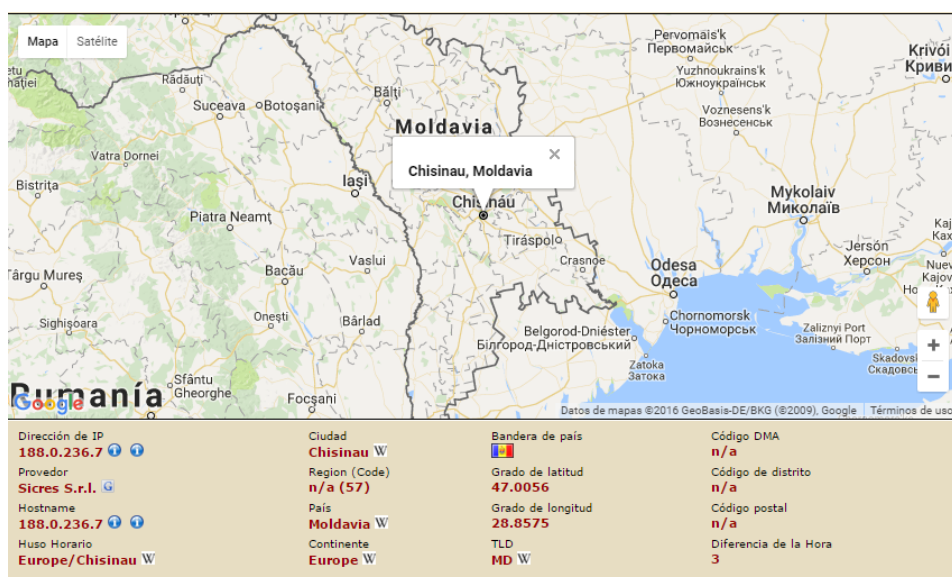


Ilustración 19. Geolocalización de la IP 188.0.236.7

15. REFERENCIAS

- <http://www.shadowexplorer.com2/>
- <https://blog.kaspersky.com/cryptxxx-ransomware/11939/>
- <http://www.bleepingcomputer.com/virus-removal/cryptxxx-ransomware-help-information>

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="cf422fca-efcc-4038-8dc9-
85fef0241b13" last-modified="2016-08-14T20:05:57"
xmlns="http://schemas.mandiant.com/2010/ioc">

  <short_description>Ransom.CryptXXX</short_description>

  <description>IOC para detectar el codigo Ransom.CryptXXX</description>

  <authored_by>CCN-CERT</authored_by>

  <authored_date>2016-08-14T19:53:08</authored_date>

  <links />

  <definition>

    <Indicator operator="OR" id="a1d21e59-e59c-4687-9817-b768fed61288">
```

² <http://www.shadowexplorer.com>

```

<IndicatorItem id="8c6c6383-a6c1-4d2e-81b8-40e0fa636e83" condition="is">
  <Context      document="RouteEntryItem"      search="RouteEntryItem/Destination"
type="mir" />
  <Content type="IP">188.0.236.7</Content>
</IndicatorItem>
<IndicatorItem id="93a5ecb6-bdff-4de9-b113-f130b817388b" condition="is">
  <Context document="FileItem" search="FileItem/Md5sum" type="mir" />
  <Content type="md5">d01fd2bb8c6296d51be297978af8b3a1</Content>
</IndicatorItem>
<IndicatorItem id="ede92705-e22a-42a5-ad81-e5091c13a26f" condition="is">
  <Context document="FileItem" search="FileItem/Md5sum" type="mir" />
  <Content type="md5">ae06248ab3c02e1c2ca9d53b9a155199</Content>
</IndicatorItem>
<IndicatorItem id="df0a1806-926f-44d4-9053-c458409e11ab" condition="is">
  <Context      document="ProcessItem"      search="ProcessItem/HandleList/Handle/Name"
type="mir" />
  <Content type="string">FX1112</Content>
</IndicatorItem>
<IndicatorItem id="f708b6e3-6cf6-4314-bacb-c9767932e278" condition="is">
  <Context      document="FileItem"
search="FileItem/PEInfo/Exports/ExportedFunctions/string" type="mir" />
  <Content type="string">XXS0S</Content>
</IndicatorItem>
<IndicatorItem id="1b959aac-c9b2-4538-9172-b3b9398cf99a" condition="is">
  <Context      document="FileItem"
search="FileItem/PEInfo/Exports/ExportedFunctions/string" type="mir" />
  <Content type="string">XXS1S</Content>
</IndicatorItem>
<IndicatorItem id="f7740d9c-7258-469b-9881-6ea70c5067cb" condition="is">
  <Context      document="FileItem"
search="FileItem/PEInfo/Exports/ExportedFunctions/string" type="mir" />
  <Content type="string">XXS2S</Content>
</IndicatorItem>
<IndicatorItem id="8e3933d7-e796-4ba7-a913-7ab2886eb631" condition="is">
  <Context      document="FileItem"
search="FileItem/PEInfo/Exports/ExportedFunctions/string" type="mir" />
  <Content type="string">XXS3S</Content>
</IndicatorItem>
<IndicatorItem id="370e6fd0-6a3e-4fa7-b6ee-a598e1eabc26" condition="is">
  <Context      document="FileItem"
search="FileItem/PEInfo/Exports/ExportedFunctions/string" type="mir" />
  <Content type="string">XXS4S</Content>
</IndicatorItem>
</Indicator>
</definition>

```

</ioc>

16.2 YARA

```

rule Ransom_CryptXXX_Dropper
{
    /*
        Regla para detectar el dropper de Ransom.CryptXXX con MD5
        d01fd2bb8c6296d51be297978af8b3a1
    */
    meta:
        description = "Regla para detectar RANSOM.CRYPTXXX"
        author      = "CCN-CERT"
        version     = "1.0"

    strings:
        $a = { 50 65 31 57 58 43 46 76 59 62 48 6F 35 }
        $b = { 43 00 3A 00 5C 00 42 00 49 00 45 00 52 00 5C 00 51 00 6D 00 6B 00 4E
00 52 00 4C 00 46 00 00 }
        condition:
            all of them
    }

rule Ransom_CryptXXX_Real
{
    /*
        Regla para detectar el codigo Ransom.CryptXXX fuera del dropper con MD5
        ae06248ab3c02e1c2ca9d53b9a155199
    */
    meta:
        description = "Regla para detectar Ransom.CryptXXX original"
        author      = "CCN-CERT"
        version     = "1.0"

    strings:
        $a = { 52 59 47 40 4A 41 59 5D 52 00 00 00 FF FF FF FF }
        $b = { 06 00 00 00 52 59 47 40 40 5A 00 00 FF FF FF FF }
        $c = { 0A 00 00 00 52 5C 4B 4D 57 4D 42 4B 5C 52 00 00 }
        $d = { FF FF FF FF 0A 00 00 00 52 5D 57 5D 5A 4B 43 70 }
        $e = { 3F 52 00 00 FF FF FF FF 06 00 00 00 52 4C 41 41 }
        $f = { 5A 52 00 00 FF FF FF FF 0A 00 00 00 52 5C 4B 4D }
        $g = { 41 58 4B 5C 57 52 00 00 FF FF FF FF 0E 00 00 00 }
        $h = { 52 2A 5C 4B 4D 57 4D 42 4B 20 4C 47 40 52 00 00 }
        $i = { FF FF FF FF 0A 00 00 00 52 5E 4B 5C 48 42 41 49 }
        $j = { 5D 52 00 00 FF FF FF FF 05 00 00 00 52 4B 48 47 }

```

```
$k = { 52 00 00 00 FF FF FF FF 0C 00 00 00 52 4D 41 40 }
$l = { 48 47 49 20 43 5D 47 52 00 00 00 00 FF FF FF FF }
$m = { 0A 00 00 00 52 5E 5C 41 49 5C 4F 70 3F 52 00 00 }
$n = { FF FF FF FF 0A 00 00 00 52 5E 5C 41 49 5C 4F 70 }
$o = { 3C 52 00 00 FF FF FF FF 08 00 00 00 52 49 41 41 }
$p = { 49 42 4B 52 00 00 00 00 FF FF FF FF 06 00 00 00 }
$q = { 52 5A 4B 43 5E 52 00 00 FF FF FF FF 08 00 00 00 }
$v = { 52 48 3A 4C 4D 70 3F 52 00 00 00 00 FF FF FF FF }
$w = { 0A 00 00 00 52 4F 42 42 5B 5D 4B 70 3F 52 00 00 }
$x = { FF FF FF FF 0A 00 00 00 52 5E 5C 41 49 5C 4F 70 }
$y = { 3F 52 00 00 FF FF FF FF 0A 00 00 00 52 5E 5C 41 }
$z = { 49 5C 4F 70 3C 52 00 00 FF FF FF FF 09 00 00 00 }
$aa = { 52 4F 5E 5E 4A 4F 5A 4F 52 00 00 00 FF FF FF FF }
$ab = { 0A 00 00 00 52 5E 5C 41 49 5C 4F 70 3D 52 00 00 }
$ac = { FF FF FF FF 08 00 00 00 52 5E 5B 4C 42 47 4D 52 }

condition:
    all of them
}
```