

SIN CLASIFICAR



Informe Código Dañino CCN-CERT ID-09/16

Ransom.Locky

Octubre de 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 VERSIONES DEL CÓDIGO DAÑINO	6
3.1.1 PRIMERA VERSIÓN	6
3.1.2 SEGUNDA VERSIÓN	7
3.1.3 TERCERA VERSIÓN	7
3.1.4 CUARTA VERSIÓN	8
3.1.5 QUINTA VERSIÓN	8
3.1.6 SEXTA VERSIÓN	8
3.1.7 SÉPTIMA VERSIÓN	8
3.2 EXTENSIONES A CIFRAR	9
3.3 EXTENSIÓN AÑADIDA A LOS ARCHIVOS CIFRADOS	10
3.4 ARCHIVOS DE RESCATE	10
4. CARACTERÍSTICAS DEL CÓDIGO DAÑINO	12
5. DETALLES GENERALES	13
6. PROCEDIMIENTO DE INFECCIÓN	14
7. CARACTERÍSTICAS TÉCNICAS	15
8. CIFRADO Y OFUSCACIÓN	22
9. PERSISTENCIA EN EL SISTEMA	26
10. CONEXIONES DE RED	27
10.1 PRIMERA VERSIÓN DEL ALGORITMO DGA	28
10.2 SEGUNDA VERSIÓN DEL ALGORITMO DGA	29
11. ARCHIVOS RELACIONADOS	30
12. DETECCIÓN	30
12.1 HERRAMIENTAS DEL SISTEMA	30
12.2 MANDIANT	31
13. DESINFECCIÓN	32
14. VACUNA CONTRA EL CÓDIGO DAÑINO	33
14.1 CHEQUEO DEL IDIOMA DEL SISTEMA	33
14.2 ENTRADA "COMPLETED" E "ID" EN EL REGISTRO	33

14.3	MODIFICACIÓN DE LOS PERMISOS DE ACCESO AL REGISTRO	34
14.4	CREACIÓN DE CLAVE RSA PÚBLICA CORRUPTA	34
14.5	CREACIÓN DE CLAVE RSA PÚBLICA CON VALOR PRE-CALCULADO	35
15.	INFORMACIÓN DEL ATACANTE	35
15.1	78.40.108.39	35
15.1.1	GEOLOCALIZACIÓN	36
15.2	91.195.12.131	36
15.2.1	GEOLOCALIZACIÓN	37
15.3	149.154.157.14	37
15.3.1	GEOLOCALIZACIÓN	38
15.4	151.236.14.51	38
15.4.1	GEOLOCALIZACIÓN	39
15.5	37.235.53.18	39
15.5.1	GEOLOCALIZACIÓN	40
15.6	51.254.240.48	40
15.6.1	GEOLOCALIZACIÓN	41
15.7	91.219.29.41	41
15.7.1	GEOLOCALIZACIÓN	42
15.8	185.82.216.55	42
15.8.1	GEOLOCALIZACIÓN	43
15.9	217.12.223.83	43
15.9.1	GEOLOCALIZACIÓN	44
16.	REFERENCIAS	44
17.	REGLAS DE DETECCIÓN	45
17.1	INDICADOR DE COMPROMISO – IOC	45
17.2	YARA	46

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.Locky**", 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.

En las primeras dos versiones del código dañino se ha distribuido mayoritariamente mediante documentos ofimáticos de tipo Word con "macros" dañinas, que eran las encargadas de descargar el código dañino de Internet y ejecutarlo. Otro medio de distribución del código dañino fue utilizar servidores comprometidos, en los cuales se alojó un "Exploit Kit", el cual aprovechaba alguna vulnerabilidad en los navegadores de los visitantes de páginas web ahí alojadas para descargar el código dañino y ejecutarlo en sus sistemas.

Desde la tercera versión el método de distribución es mediante archivos *JavaScript* que están fuertemente ofuscados y que, aparte de ser los encargados de la descarga del binario y su posterior ejecución, realizan una serie de acciones necesarias para que el código dañino pueda funcionar en el sistema comprometido.

A partir de inicios de agosto de 2016 el código *JavaScript* cambió el modo en el que descifra los binarios, al igual que ya no se requiere la acción de enviar un parámetro al "dropper" al crear el código dañino.

3. INFORMACIÓN DE VERSIONES DEL CÓDIGO DAÑINO

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

3.1 VERSIONES DEL CÓDIGO DAÑINO

El código dañino se ha ido modificando desde su aparición habiéndose encontrado siete versiones, cuyas características más importantes se enumeran a continuación:

3.1.1 PRIMERA VERSIÓN

- Su primera aparición conocida se remonta al día 16 de febrero del 2016.
- Su método de difusión es mediante correos de *Spam/Phishing* con un archivo adjunto ofimático. Este documento lleva "macros" dañinas que efectúan la descarga y posterior ejecución del código dañino en el sistema.
- Posee un valor embebido que usa como identificador para saber a quién tiene que pagar un porcentaje de la cantidad recaudada por el secuestro de los archivos. El identificador de afiliado está a 1 o 2. La gran mayoría de muestras tienen el valor 1.
- Crea una entrada en el registro llamada "Locky" en la rama de "HKEY_CURRENT_USER\SOFTWARE".

- El tamaño del código dañino varía entre 95.744 bytes y 98.304 bytes.
- El algoritmo DGA (*Domain Generation Algorithm*) permite generar dominios de forma pseudo-aleatoria a los que el código dañino intentará conectar. Para ello utiliza dos claves embebidas junto con la hora y fecha del sistema.
- El algoritmo DGA puede llegar a generar hasta 6 dominios únicos en dos días.

3.1.2 SEGUNDA VERSIÓN

- Su aparición data de primeros de marzo del 2016.
- Su método de difusión continúa basándose en los envíos de correo mediante *Spam/Phishing*, sin embargo ya no se adjunta un documento ofimático, sino un archivo *JavaScript* ofuscado que es el encargado de descargar el código dañino e infectar el sistema.
- El tamaño del código dañino es de 106.496 bytes.
- Al igual que la versión anterior, se crea una entrada en el registro llamada "Locky" en la rama de "HKEY_CURRENT_USER\SOFTWARE\".
- El identificador de afiliado tiene los valores 3 o 4.
- El algoritmo del DGA ya no utiliza únicamente la fecha y hora del sistema, ahora usa valores embebidos junto con la fecha y hora. Puede llegar a generar hasta 8 dominios únicos en dos días.

3.1.3 TERCERA VERSIÓN

- Su aparición se estima que data de finales de marzo del 2016. Es, sin lugar a dudas, la respuesta de los creadores del código dañino a las vacunas creadas para prevenir la infección de esta amenaza sobre nuevas víctimas.
- El tamaño del código dañino varía entre los 110.592 bytes ("110kB") y 122.880 bytes ("120kB").
- El código dañino posee una nueva capa de ofuscación para ralentizar el proceso de análisis y evitar herramientas de volcado de memoria automatizadas.
- Este nuevo sistema "remapea" el código dañino a una región creada dinámicamente, para dificultar el análisis del código y evitar herramientas de volcado de procesos.
- En alguna de las nuevas muestras del código dañino de esta versión, las direcciones IP embebidas son copiadas a un buffer creado dinámicamente en el arranque de la aplicación. Esto se realiza para intentar ofuscar más el código.

- Establece comunicación a una dirección distinta en el servidor C2 (Mando y Control) que las dos versiones anteriores.
- A diferencia de las anteriores versiones del código dañino, no se crea una entrada en el registro en la clave "HKEY_CURRENT_USER\Software\Locky". En su lugar se crea una entrada con un valor que varía dependiendo de la muestra ejecutada, por ejemplo, "HKEY_CURRENT_USER\Software\029CjNI6". En dicha entrada se crean tres valores con nombres aleatorios donde se guarda la clave pública RSA obtenida del servidor C2, el texto a mostrar para recuperar los archivos y la palabra "YES". Los valores están cifrados con un algoritmo propietario.
- La imagen creada y puesta como fondo de escritorio es ligeramente distinta a la de las dos primeras versiones del código dañino.

3.1.4 CUARTA VERSIÓN

- Su aparición se data entre finales de mayo del 2016 y principios de junio del 2016.
- El tamaño del código dañino varía entre los 165.888 bytes ("162kB") y 167.898 bytes ("163kB").
- Esta versión del código dañino emplea como vector de infección archivos *JavaScript* que descargan, descifran y ejecutan los binarios que contienen el código dañino.
- Establece comunicación a un servidor C2 distinto de las tres versiones anteriores.
- La imagen creada y puesta como fondo de escritorio es ligeramente distinta a las anteriores.

3.1.5 QUINTA VERSIÓN

- Su aparición data de mediados de junio del 2016.
- Su cambio principal es la nueva extensión que utiliza: ".zepto".

3.1.6 SEXTA VERSIÓN

- Su aparición data de principios de agosto de 2016.
- Sus características son muy similares a la quinta versión, cambiando el código de descifrado con el que usa en cierta parte de su código.

3.1.7 SÉPTIMA VERSIÓN

- Su aparición data de mediados de agosto de 2016.
- Sus características son muy similares a la versión anterior sin embargo el código *JavaScript* utilizado para su descarga y descifrado sufrió grandes cambios.

- El "dropper" usado no es el mismo que se usó en la quinta y sexta versión.
- Se crea un archivo llamado "reper.log" en la carpeta desde la que se haya ejecutado el "dropper". Dicho archivo tiene la fecha en la que se ejecutó y no es eliminado cuando el "dropper" se elimina del disco.

3.2 EXTENSIONES A CIFRAR

El código dañino, en todas sus versiones, cifra todos los archivos que tengan alguna de las siguientes extensiones:

wallet.dat	.xlt	.ms11 (security copy)	.h	.7z
.key	.xlw	.lay	.js	.gz
.crt	.slk	.lay6	.vb	.tgz
.csr	.xlsb	.asc	.vbs	.tar
.p12	.xlsm	.onetoc2	.pl	.bak
.pem	.xlsx	.pst	.dip	.tbk
.doc	.xltn	.001	.dch	.tar.bz2
.odt	.xltx	.002	.sch	.paq
.ott	.wk1	.003	.brd	.arc
.sxw	.wks	.004	.cs	.aes
.stw	.123	.005	.asp	.gpg
.ppt	.wb2	.006	.rb	.vmx
.xls	.odp	.007	.java	.vmdk
.pdf	.otp	.008	.jar	.vdi
.rff	.sxi	.009	.class	.qcow2
.uot	.sti	.010	.pl	.mp3
.csv	.pps	.011	.sh	.wav
.txt	.pot	.sqlite3	.bat	.swf
.xml	.sxd	.sqlitedb	.cmd	.fla
.3ds	.std	.sql	.psd	.wmv
.max	.pptm	.mdb	.nef	.mpg
.3dm	.pptx	.db	.tiff	.vob
.dot	.potm	.dbf	.tif	.mpeg
.docx	.potx	.odb	.jpg	.asf
.docm	.uop	.frm	.jpeg	.avi
.dotx	.odg	.myd	.cgm	.mov
.dotm	.otg	.myi	.raw	.mp4
.602	.sxm	.ibd	.gif	.3gp
.hwp	.mml	.mdf	.png	.mkv
.ods	.docb	.ldf	.bmp	.3g2

.ots	.ppam	.php	.svg	.flv
.sxc	.ppsx	.c	.djvu	.wma
.stc	.ppsm	.cpp	.djv	.mid
.dif	.sldx	.pas	.zip	.m3u
.xlc	.sldm	.asm	.rar	.m4u
.xlm	.ms11			

3.3 EXTENSIÓN AÑADIDA A LOS ARCHIVOS CIFRADOS

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

.locky

Desde la quinta versión del código dañino cambia la extensión a los archivos cifrados con la siguiente:

.zepto

3.4 ARCHIVOS DE RESCATE

El código dañino crea una serie de ficheros en el sistema comprometido con información sobre el secuestro y formas de recuperar los archivos cifrados según su versión tal y como muestran las ilustraciones siguientes:

La imagen del rescate en las dos primeras versiones es la misma.

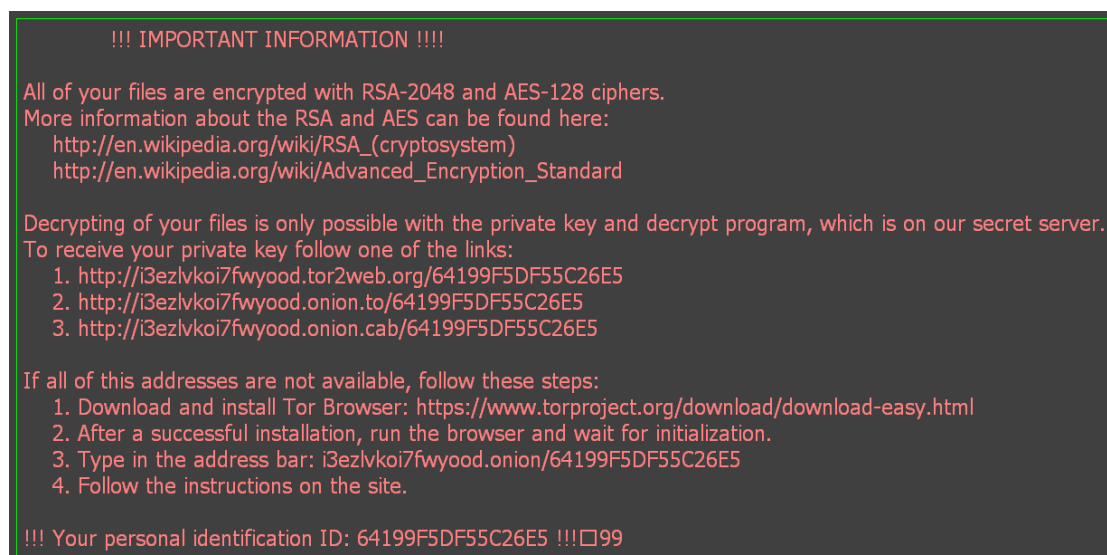


Ilustración 1. Información del secuestro de los archivos en el sistema

La imagen del rescate de la tercera versión del código dañino es ligeramente distinta a la de las dos primeras versiones:

```

*-~_|~-.+_~*
=$~_+$*$
!!! IMPORTANT INFORMATION !!!!

All of your files are encrypted with RSA-2048 and AES-128 ciphers.
More information about the RSA and AES can be found here:
  http://en.wikipedia.org/wiki/RSA_(cryptosystem)
  http://en.wikipedia.org/wiki/Advanced_Encryption_Standard

Decrypting of your files is only possible with the private key and decrypt program, which is on our secret server.
To receive your private key follow one of the links:
  1. http://25z5g623wpqpdwis.tor2web.org/C4D48B539F26864B
  2. http://25z5g623wpqpdwis.onion.to/C4D48B539F26864B
  3. http://25z5g623wpqpdwis.onion.cab/C4D48B539F26864B

If all of this addresses are not available, follow these steps:
  1. Download and install Tor Browser: https://www.torproject.org/download/download-easy.html
  2. After a successful installation, run the browser and wait for initialization.
  3. Type in the address bar: 25z5g623wpqpdwis.onion/C4D48B539F26864B
  4. Follow the instructions on the site.

!!! Your personal identification ID: C4D48B539F26864B !!!
_|*_$+_$.|~*
$~_|.+*~+$+
--+~-.*.
|~_~$~-.+.
□~_

```

Ilustración 2. Fondo de escritorio de la tercera versión del código dañino

En la a partir de la cuarta versión la imagen también está ligeramente modificada:

```

.=+$.
|=|*~
!!! IMPORTANT INFORMATION !!!!

All of your files are encrypted with RSA-2048 and AES-128 ciphers.
More information about the RSA and AES can be found here:
  http://en.wikipedia.org/wiki/RSA_(cryptosystem)
  http://en.wikipedia.org/wiki/Advanced_Encryption_Standard

Decrypting of your files is only possible with the private key and decrypt program, which is on our secret server.
To receive your private key follow one of the links:
  1. http://mphtadhci5mrdlju.tor2web.org/845B9F15C35F5112
  2. http://mphtadhci5mrdlju.onion.to/845B9F15C35F5112

If all of this addresses are not available, follow these steps:
  1. Download and install Tor Browser: https://www.torproject.org/download/download-easy.html
  2. After a successful installation, run the browser and wait for initialization.
  3. Type in the address bar: mphtadhci5mrdlju.onion/845B9F15C35F5112
  4. Follow the instructions on the site.

!!! Your personal identification ID: 845B9F15C35F5112 !!!
|~=$~+$+~+~_
_$.+~|~_~+*~+
$$$~+
□$$$

```

Ilustración 3. Fondo de escritorio a partir de la cuarta versión

Los archivos creados en la primera y segunda versión, se engloban en la siguiente tabla:

```
_Locky_recover_instructions.txt  
_Locky_recover_instructions.html  
_Locky_recover_instructions.bmp
```

Los archivos de texto son creados en todas las carpetas en las cuales el código dañino cifra algún archivo. El archivo ".html" y el archivo ".bmp" son creados también en el escritorio del usuario del sistema comprometido.

En la tercera y cuarta versión del código dañino se crean una serie de archivos con un nombre diferentes al de las primeras dos versiones:

```
_HELP_instructions.txt  
_HELP_instructions.bmp
```

Desde la quinta versión del código dañino los archivos creados difieren ligeramente. En cada carpeta en donde haya cifrado algún archivo creará un único archivo con el siguiente nombre:

```
_<valor_númeroico_tres_digitos>_HELP_instructions.html
```

Además, se crean en el escritorio dos archivos más:

```
_HELP_instructions.html  
_HELP_instructions.bmp
```

4. CARACTERÍSTICAS DEL CÓDIGO DAÑINO

El código dañino examinado posee las siguientes características:

- Carga el código dañino en el sistema.
- Modifica archivos del sistema, cifrándolos y borrando los originales. Tras ello pide un rescate para recuperar los originales.
- Conecta con un C2 preestablecido en su código.
- Accede a los recursos de red disponibles, estén montados o no, para enumerar y cifrar archivos en ellos. En el caso de que no estén montados procede a crear un punto de montaje y crear una conexión.
- Enumera todos los archivos del sistema que cumplan un patrón (la extensión) y que no estén en determinados directorios, y los cifra.
- Modifica el registro del sistema para asegurar persistencia y guardar información importante acerca de su ejecución y funciones. A partir de la cuarta versión no se guarda ninguna información en el registro de Windows así como tampoco se crean entradas de persistencia en las muestras analizadas.

5. DETALLES GENERALES

Las muestras analizadas se corresponden con las siguientes firmas MD5:

```

07e1469a71cbf40565d2dad8bf1e2264
777034d6c401ed6823189fa46f8f2b2f
e96dad009437ca774035ffd73708bd3e
bf60e7f30915e5b9a150fe0b4e4de72e
a4922a9d6ec69e6ae050d6174f1bf5a2
16bf6b671c586b121f9708074c2e2c7d
3e7895f1a3049c21a3592927ea53109a
b550c37041614c944eb28b35aa8be452
  
```

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

Se ha podido observar que la fecha interna de creación del programa ha sido alterada con valores antiguos como en el caso de una de las muestras que data del 12 de enero del 2010.

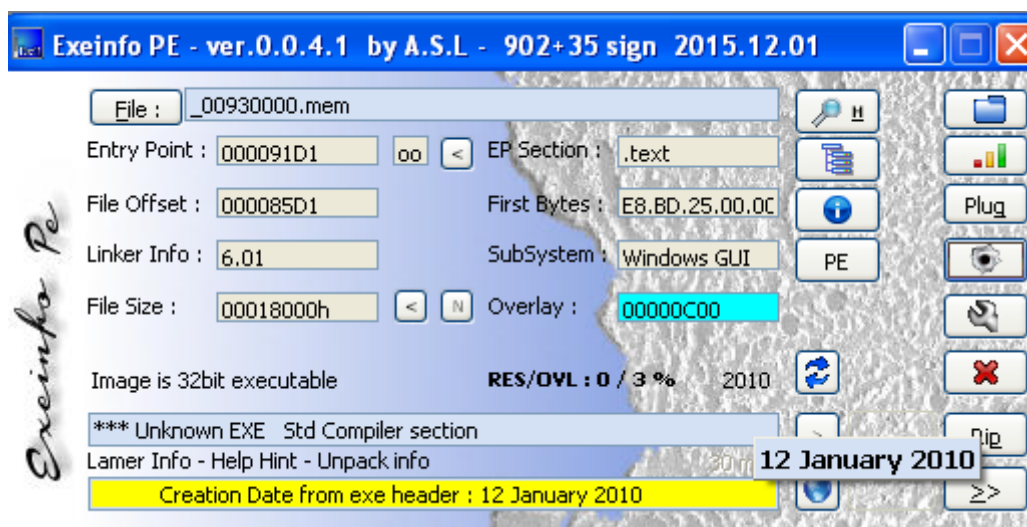


Ilustración 4. Información de la muestra 07e1469a71cbf40565d2dad8bf1e2264

Las otras tres muestras analizadas datan de las fechas siguientes:

bf60e7f30915e5b9a150fe0b4e4de72e	24 de diciembre de 2007
777034d6c401ed6823189fa46f8f2b2f	21 de agosto de 2010
07e1469a71cbf40565d2dad8bf1e2264	21 de agosto de 2010

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:

- Abriendo un documento Word, típicamente enviado a través de un correo electrónico, teniendo habilitadas las macros o autorizando que éstas se ejecuten y así se produzca la descarga y ejecución del código dañino.
- Ejecutando un código *JavaScript* ofuscado, también enviado a través de un correo electrónico, que produzca la descarga del código dañino y su ejecución.
- Visitando una página web comprometida con un "Exploit Kit", usando un navegador no actualizado que permita la ejecución del código dañino directamente, al explotar alguna vulnerabilidad del navegador.
- Ejecutando el código dañino si se realizó una descarga por una red P2P.

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

- El código dañino descargado por cualquiera de los medios indicados anteriormente es un "dropper". Este tipo de programas tienen la función de descifrar en memoria el código dañino final, el "ransomware" en este caso, y ejecutarlo.
- Se copia a sí mismo en la carpeta %TEMP% del sistema y se vuelve a ejecutar desde esa ubicación.
- Enumera todos los discos del sistema y unidades de red.
- Borra los "Shadow Volume" del sistema.
- Se conecta con su dominio C2 para obtener la clave de cifrado.
- Busca archivos con determinadas extensiones en todos los discos y unidades enumeradas previamente para cifrarlos.
- Muestra información acerca del secuestro de los archivos y los modos para acceder a la información cifrada.

Desde la tercera versión, el procedimiento varía un poco:

- Se emplea como vector de infección archivos *JavaScript* que descargan y ejecutan los binarios que contienen el código dañino.
- El código dañino es descargado de forma cifrada por parte del *JavaScript* para evitar sistemas IDS/IPS. Una vez descargado, el *JavaScript* procede al descifrado del binario en disco ya como ejecutable.
- El código dañino descargado no es funcional directamente y necesita que el código *JavaScript* lo ejecute con un parámetro numérico de 8 bits, que es usado para el proceso de descifrado por el "dropper". En el caso

de que no exista el parámetro en la ejecución, o no sea el esperado, el "dropper" producirá un error finalizando su ejecución.

- En las muestras analizadas se ha podido comprobar que los valores "123" y "321" permiten ejecutar el "dropper" sin problemas pudiendo obtener el código dañino final.
- Al ser una clave de 8 bits, sólo se permite un valor comprendido entre 0 a 255, y es por ello que sólo se considera el valor de 8 bits del registro. Por ejemplo, si una muestra tiene como valor de descifrado en hexadecimal "0x41", sería como válido un parámetro en decimal de "321", "1601" o "2113". Es por ello que calcular la clave a base de fuerza bruta es trivial.
- En la sexta versión el valor necesario para un funcionamiento correcto cambió al "323" aunque sigue teniendo el mismo problema que las anteriores dos versiones de múltiples valores. Por ejemplo, en este caso también es válido el valor "67".
- Lo anteriormente indicado ya no es utilizado en la séptima versión del código dañino. En esta versión el sistema de descifrado del binario en el JavaScript cambió completamente, aparte de que el nuevo "dropper" no requiere ningún valor para descifrar parte de su código.

7. CARACTERÍSTICAS TÉCNICAS

El código dañino posee una primera capa de protección mediante un "dropper", que será el encargado de descifrar el código dañino final y ponerlo en ejecución, camuflándose como una aplicación normal. En las muestras analizadas tenía distintos iconos y los datos de la aplicación. En el proceso final del "dropper", descifra en memoria el código dañino y procede a ejecutarlo, tras ello, el propio "dropper" se borra del disco.

Desde la tercera versión hasta la sexta versión del código dañino, se ha añadido una capa extra de cifrado en su inicio que es invocada desde el JavaScript que lo descarga. En el momento de ser invocado para su ejecución, además, se le pasa como parámetro una clave de descifrado desde el JavaScript para dificultar todavía más el análisis.

En las muestras pertenecientes a las versiones que van desde la tercera a la séptima versión del código dañino, se usa una nueva capa de ofuscación antes de sacar el código dañino final en sí. Según la muestra, la complejidad puede ser mayor o menor en esta parte, debido al código utilizado y las llamadas de sistema que realice.

El código dañino posee una estructura de configuración interna según la cual, mediante dos "flags", le indica si debe realizar dos operaciones en su ejecución.

La primera de las operaciones es comprobar si el sistema está en lenguaje ruso. Si es así, se elimina y no realiza ningún tipo de cifrado ni actividad en el sistema.

La segunda de las operaciones es dormir el hilo principal por un tiempo definido por el valor establecido en la configuración multiplicado por "0x3E8". Esto lo realiza

para intentar impedir que sistemas de análisis automáticos puedan analizar su comportamiento.

Esta estructura también guarda valores para indicarle al código dañino si debe copiarse al directorio temporal del sistema (tal y como se verá posteriormente) o si debe crear entradas de persistencia en el registro. Estos dos últimos valores siempre tienen valor "1" en las muestras analizadas. Otro valor importante en esta estructura es una semilla que será usada posteriormente en el algoritmo de generación de dominios. Este campo sólo se aplica desde la tercera versión del código dañino.

Tras arrancar, el código dañino obtiene la dirección de memoria de la función "Wow64DisableWow64FsRedirection". Esta función permite deshabilitar la redirección existente en plataformas de 64 bits para que las aplicaciones de 32 bits sean redirigidas al directorio nativo donde están sus librerías, "%WINDIR%\SysWOW64", y no al que intentan acceder de forma natural, "%WINDIR%\System32", que contiene las librerías de 64 bits¹. Si dicha función se encuentra en el sistema, procede a invocarla para deshabilitar la redirección de archivos.

Posteriormente, procede a acceder a su propio proceso para obtener el "token" que indica los privilegios que posee el proceso para realizar acciones. Una vez obtenido el "token", procede a deshabilitar la virtualización, si estuviera activa para ese mismo proceso.

00404073	- 50	push	eax		phToken
00404074	- 33DB	xor	ebx, ebx		
00404076	- 68 80000000	push	80		DesiredAccess = TOKEN_ADJUST_DEFAULT
00404078	- 895D AC	mov	[ebp-54], ebx		
0040407E	- FF15 E010410	call	[&KERNEL32.GetCurrentProcess]		[GetCurrentProcess
00404084	- 50	push	eax		hProcess
00404085	- FF15 2010410	call	[&ADUAPI32.OpenProcessToken]		OpenProcessToken
00404088	- 85C0	test	eax, eax		
0040408D	- 74 1A	je	short 004040A9		
0040408F	- 6A 04	push	4		DataSize = 4
00404091	- 8D45 AC	lea	eax, [ebp-54]		
00404094	- 50	push	eax		Data
00404095	- 6A 18	push	18		InfoClass = 24.
00404097	- FF75 D8	push	dword ptr [ebp-20]		hToken
0040409A	- FF15 1C10410	call	[&ADUAPI32.SetTokenInformation]		SetTokenInformation

Ilustración 5. Deshabilitando la virtualización para el proceso del código dañino

La siguiente acción del código dañino depende de su versión. Las últimas versiones del código dañino obtienen el lenguaje del sistema. En el caso de que sea ruso, asigna los atributos del propio archivo del código dañino a "NORMAL" y mueve ese mismo ejecutable a la carpeta %TEMP%, con un nombre aleatorio que comience siempre por "sys", utilizando la función "GetTempFileName". Una vez movido a la nueva ubicación procede a su borrado mediante un comando y finaliza su ejecución:

```
cmd /C del /Q /F <archivo_temporal_con_prefijo_sys>
```

Tanto en el caso de que el idioma no sea ruso o que el código dañino sea de la primera versión, su primera acción es generar un identificador, que es el hash MD5 del GUID obtenido desde el punto de montaje de la unidad donde se encuentre instalado el sistema operativo afectado.

¹ <https://msdn.microsoft.com/es-es/library/windows/desktop/aa365743>

Para poder realizar esta acción, el código dañino obtiene el directorio de Windows del sistema comprometido mediante la función "GetWindowsDirectory", le añade el carácter "/" y obtiene el GUID del punto de montaje de ella, usando la función "GetVolumeNameForVolumeMountPoint" en el caso de que se pueda obtener. Tras obtener el GUID, el código dañino busca el carácter "{" y el carácter "}", para delimitar la cadena del GUID desde su principio a su final y sobre esa cadena calcula el hash MD5.

El código dañino convierte el hash obtenido a una representación en ASCII, y tras esta conversión, toma los primeros 16 caracteres como identificadores únicos del sistema comprometido. Se muestra, a continuación, el pseudo-código de este proceso.

```
windir = GetWindowsDirectory;  
mount_point_name = GetVolumeNameForVolumeMountPoint(windir);  
guid_mount = get_guid(mount_point_name);  
md5_guid = md5calc(guid_mount)  
id = md5_guid.uppercase.substring(0,16);
```

En este punto el código dañino no comprueba si el resultado de la llamada a la función "GetVolumeNameForVolumeMountPoint" resultó con un GUID válido. En algunos casos la función puede fallar devolviendo el error 1126. En ese caso, el código dañino busca igualmente los caracteres del inicio y final del GUID que, obviamente, no va a encontrar, y calcula el hash de lo que tenga en la dirección de memoria que retornó la función. El valor que reside en esa dirección de memoria, puede ser una nueva dirección de memoria o directamente información "basura" de la pila del procesador. Al ocurrir esto, y dependiendo de la muestra ejecutada, puede ocurrir que la dirección de memoria o la información de la pila sea distinta, produciendo otro identificador distinto.

Posteriormente, crea una entrada en el registro que usará como punto donde guardar distintas informaciones necesarias para su ejecución.

```
[HKEY_CURRENT_USER\Software\Locky]
```

En dicha entrada procede a buscar determinados campos, los cuales, si existen, indican que el código dañino ya se había ejecutado previamente en el sistema. Estas entradas son:

- **id:** el identificador único generado previamente.
- **pubkey:** en formato binario, es la clave pública RSA para cifrar las claves AES que se usan para cifrar los archivos en el sistema comprometido.
- **paytext:** incluye el texto a mostrar del secuestro de los archivos, la forma de recuperarlos y el idioma del sistema. Este texto será el usado para generar el contenido de la imagen que se pondrá en el escritorio.

- **completed:** valor DWORD con el valor a "1" si el código dañino llegó a ejecutarse de forma completa en una ejecución previa. Esta entrada es comprobada y en el caso de que exista con un valor a "1", el código dañino comprueba el valor del campo "id" del registro con el identificador único generado previamente. En el caso de que coincidan, procede al borrado del código dañino y a finalizar su ejecución, en caso contrario, la ejecución continúa. Esto indica que el código dañino previene de posteriores ejecuciones en un sistema en el que haya sido ejecutado previamente.

En la tercera versión del código dañino, no se crea exactamente esa entrada de registro indicada anteriormente sustituyéndola en su lugar por una entrada con un valor aleatorio que varía entre cada muestra.

[HKEY_CURRENT_USER\Software\<valor aleatorio>]

En dicha entrada se guardan los tres elementos tal y como ocurría con las versiones anteriores del código dañino.

Las entradas tienen un nombre formado por caracteres aleatorios, si bien el resultado es constante en la misma muestra. Una de las entradas guarda 114 bytes de información, que es la clave RSA pública obtenida del servidor C2. Otra entrada incluye el texto a mostrar pidiendo el rescate de los archivos en el idioma del sistema comprometido obtenido del servidor C2. Ambas entradas están cifradas con un algoritmo propietario.

 (Default)	REG_SZ	(value not set)
 86Ymj06QifNkmr2	REG_BINARY	cc 6d c7 2f c5 de 04 bf e2 21 5c 5e f1 8c 4d 63 55 07 2...
 04v69ekyo	REG_BINARY	51 10 c3 8d e0 c9 05 08 4e 8c cb c5 d5 f7 ab d6 42 67 ...

Ilustración 6. Nuevas entradas del código dañino en el registro

La última entrada tiene cifrado el texto "YES".

En las versiones a partir de la cuarta no hay ninguna de estas entradas en el registro.

Posteriormente el código se copia a sí mismo en la carpeta "%TEMP%" con el nombre "svchost.exe". Una vez copiado, procede a borrar el ADS (*Alternate Data Stream*) "Zone.Identifier" del archivo recién copiado y lanza su ejecución desde esa ubicación.

El código dañino intenta obtener la siguiente información del sistema comprometido:

- Lenguaje del sistema, mediante las funciones "GetUserDefaultUILanguage"² y "GetLocaleInfo"³.

² <https://msdn.microsoft.com/en-us/library/windows/desktop/dd318137%28v=vs.85%29.aspx>

³ <https://msdn.microsoft.com/en-us/library/windows/desktop/dd318101%28v=vs.85%29.aspx>

- El identificador generado previamente.
- El dominio del equipo comprometido si estuviera en alguno.
- Versión del sistema operativo.

Address	Hex	dump	Disassembly	Comment
00402F12	>	BE E0284100	mov esi, 004128E0	ASCII "Windows XP"
00402F17	>	E9 83000000	jmp 00402FCF	
00402F1C	>	3BC3	cmp eax, ebx	
00402F1E	>	75 0A	jnz short 00402F2A	
00402F20	>	BE EC284100	mov esi, 004128EC	ASCII "Windows 2003"
00402F25	>	E9 A5000000	jmp 00402FCF	
00402F2A	>	BE FC284100	mov esi, 004128FC	ASCII "Windows 2003 R2"
00402F2F	>	E9 9B000000	jmp 00402FCF	
00402F34	>	83BD 08FFFFFF	cmp [local.62], 6	
00402F38	>	75 68	jnz short 00402FA5	
00402F3D	>	8B85 0CFFFFFF	mov eax, [local.61]	
00402F43	>	3BC3	cmp eax, ebx	
00402F45	>	75 14	jnz short 00402F5B	
00402F47	>	807D 9E 01	cmp byte ptr [ebp-62], 1	
00402F48	>	75 07	jnz short 00402F54	
00402F4D	>	BE 0C294100	mov esi, 0041290C	ASCII "Windows Vista"
00402F52	>	EB 7B	jmp short 00402FCF	
00402F54	>	BE 1C294100	mov esi, 0041291C	ASCII "Windows Server 2008"
00402F59	>	EB 74	jmp short 00402FCF	
00402F5B	>	3BC6	cmp eax, esi	
00402F5D	>	75 14	jnz short 00402F73	
00402F5F	>	807D 9E 01	cmp byte ptr [ebp-62], 1	
00402F63	>	75 07	jnz short 00402F6C	
00402F65	>	BE 30294100	mov esi, 00412930	ASCII "Windows 7"
00402F6A	>	EB 63	jmp short 00402FCF	
00402F6C	>	BE 3C294100	mov esi, 0041293C	ASCII "Windows Server 2008 R2"
00402F71	>	EB 5C	jmp short 00402FCF	
00402F73	>	83F8 02	cmp eax, 2	
00402F76	>	75 14	jnz short 00402F8C	
00402F78	>	807D 9E 01	cmp byte ptr [ebp-62], 1	
00402F7C	>	75 07	jnz short 00402F85	
00402F7E	>	BE 54294100	mov esi, 00412954	ASCII "Windows 8"

Ilustración 7. Obtención del sistema operativo comprometido

- El identificador del afiliado: en las muestras iniciales del código dañino analizadas todas tenían este valor a "1", sin embargo, en muestras posteriores, tienen este valor a 4. Cabe indicar que se han localizado muestras con valores "2" y "3".
- Si es una versión corporativa. En las muestras analizadas se pone este valor a "0".
- Si el sistema comprometido es un servidor o no.
- El número de Service Pack del sistema operativo.
- Si es un sistema de 64 bits o no.

Una vez obtenida toda la información el código dañino crea una cadena con toda la información de la siguiente manera:

- id=<identificador>
- act=<acción a realizar>, en este caso la cadena contiene el texto "getkey".
- affid=<identificador del afiliado>, las muestras analizadas ponen este valor a "1" o "4". En últimos casos con valores a "2" o "3".
- lang=<código del lenguaje del sistema>
- corp=<si es versión corporativa o no>

- serv=<si es un servidor o no>
- os=<la versión del sistema operativo>, en esta cadena cualquier espacio se ha convertido en un carácter "+".
- sp=<la versión del Service Pack del sistema>
- x64=<si es un sistema de 64 bits o no>

ASCII "id=64199F5DF55C26E5&act=getkey&affid=1&lang=en&corp=0&serv=0&os=Windows+XP&sp=3&x64=0"

Ilustración 8. Ejemplo de información obtenida del sistema comprometido

Tras haber obtenido toda esta información, el código dañino calcula un *hash* MD5 de toda la cadena. Este MD5 es añadido, tras ser convertido a su equivalente en ASCII, al principio de la cadena creada anteriormente. Posteriormente el código dañino genera "3" valores aleatorios de 32 bits cada uno y los concatena a la cadena anterior. Luego cifra toda la cadena con un algoritmo propietario.

El código dañino comprueba que tiene direcciones IP a las que conectar embebidas en su código, en el caso de no posea ninguna o no conecte con ellas, procederá a ejecutar la función de generación de dominios DGA (Domain Generation Algorithm). Dicha función es propietaria y se basa en información como la fecha del sistema y ciertos valores prefijados en el propio código. Sea con una de las direcciones IP embebidas o con un dominio generado aleatoriamente, el código dañino lo usará como servidor C2 al que conectar.

A continuación, crea una conexión a su primer servidor C2 al cual envía toda la cadena cifrada mediante una solicitud POST. El servidor le contesta con una gran cadena de bytes cifrados. El código dañino descifra esta cadena mediante un algoritmo propietario, obteniendo tras ello la clave pública RSA que será usada para cifrar los archivos del sistema. Esa clave se escribe en el registro de Windows en las entradas indicadas, para guardar esa información junto con el identificador generado inicialmente.

[HKEY_CURRENT_USER\Software\Locky\id] = <identificador_único> [HKEY_CURRENT_USER\Software\Locky\pubkey] = <clave_rsa>

Tras escribir la información en el registro, procede a solicitar al C2 el texto que tendrá que mostrar de rescate en el idioma en el que esté el sistema comprometido. Para ello crea de nuevo una cadena con el contenido:

- id=<identificador inicial>
- act=gettext
- lang=<código del idioma del sistema comprometido>

El código dañino procede a cifrar la cadena recién creada de la misma forma que cifró la anterior, y la envía al servidor C2, que le devuelve una cadena cifrada que contiene el texto a mostrar, indicando el secuestro de los archivos y los pasos a seguir. Esta cadena se almacena bajo la entrada "paytext" del registro.

```
[HKEY_CURRENT_USER\Software\Locky\paytext] = <texto_del_secuestro>
```

Además del texto, se envía un *hash* MD5 del texto que permite validar la integridad de la información recibida.

A tercera versión del código, las acciones realizadas para la obtención de la clave pública RSA y el texto a mostrar sobre el secuestro de los archivos son muy similares a las versiones anteriores. La única diferencia radica en el lugar y la forma en la que se guarda la información recibida. Tanto la clave RSA como el texto son cifrados una vez se han obtenido desde el servidor C2, se ha comprobado su integridad y se ha descifrado de la primera capa que envía el C2. Este algoritmo de cifrado es propietario. A partir de la cuarta versión el código dañino no almacena esta información en el registro sino que se tiene directamente en memoria.

Posteriormente, el código dañino procede a enumerar todos los recursos de red disponibles en el sistema comprometido. Por cada recurso encontrado, tanto si ya está montado o si puede montar él mismo, accederá a dicho recurso y comenzará un nuevo hilo, que buscará archivos objetivos y los cifrará en ese mismo recurso.

Tras ello, enumerará todas las unidades lógicas del sistema y por cada una de ellas que sea de tipo disco duro, unidad removible o disco RAM procederá a crear un hilo a la función de búsqueda y cifrado de archivos.

El código dañino intenta eliminar todos los "Shadow Volumes" que puedan existir en el sistema comprometido, mediante la ejecución del programa de Windows "vssadmin".

0006FBF0	00000000	ModuleFileName = NULL
0006FBF4	00983F48	CommandLine = "vssadmin.exe Delete Shadows /All /Quiet"
0006FBF8	00000000	pProcessSecurity = NULL
0006FBFC	00000000	pThreadSecurity = NULL
0006FC00	00000000	InheritHandles = FALSE
0006FC04	00000050	CreationFlags = CREATE_NEW_CONSOLE IDLE_PRIORITY_CLASS
0006FC08	00000000	pEnvironment = NULL
0006FC0C	00000000	CurrentDir = NULL
0006FC10	0006FC24	pStartupInfo = 0006FC24
0006FC14	0006FC68	pProcessInfo = 0006FC68

Ilustración 9. Borrado de los Shadow Volumes del sistema

El código dañino accede al registro de Windows y crea una entrada en la siguiente rama.

```
[HKEY_CURRENT_USER\Software\Microsoft\CurrentVersion\Run\locky] = [ruta_al_código_dañino].
```

Posterior al cifrado de los archivos, el código dañino dejará en el escritorio del sistema comprometido una imagen de tipo BMP, que será configurado como fondo de escritorio para que la víctima pueda ver el mensaje. Este fichero BMP muestra la información del secuestro de archivos y los pasos necesarios para poder restablecerlos. También ejecuta el "Bloc de Notas", abriendo un fichero de texto con la misma información del fondo de escritorio.

Tras finalizar el paso anterior el código dañino borrará su entrada de persistencia, aunque dejará los datos introducidos en el registro con información. Esta tabla impide que el código dañino pueda volver a ejecutarse de forma accidental.

Por último, se mueve a la carpeta %TEMP% con un nombre aleatorio para asegurar su borrado en el siguiente reinicio.

En el caso de que se visité alguna de las URL indicadas en los archivos con información de recuperación de los archivos cifrados, se puede ver una web como esta:

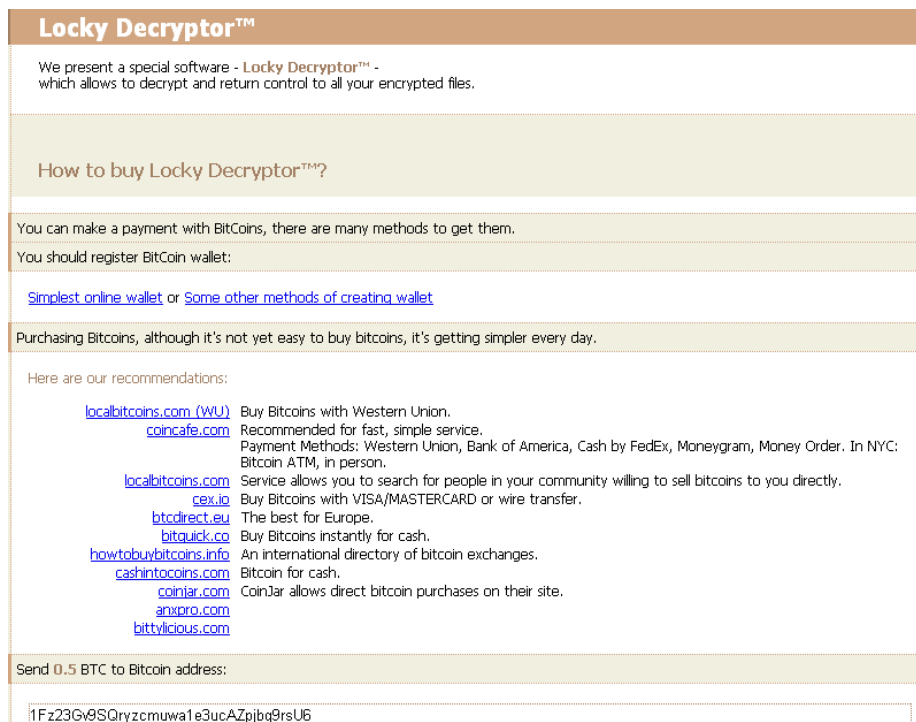


Ilustración 10. Página descifradora del código dañino

8. CIFRADO Y OFUSCACIÓN

El código dañino tiene la funcionalidad de cifrar los archivos del sistema que cumplan una serie de patrones, a través de los algoritmos AES⁴ (*Advanced Encryption Standard*) y RSA⁵ (*Rivest, Shamir y Adleman*).

⁴ https://es.wikipedia.org/wiki/Advanced_Encryption_Standard

⁵ <https://es.wikipedia.org/wiki/RSA>

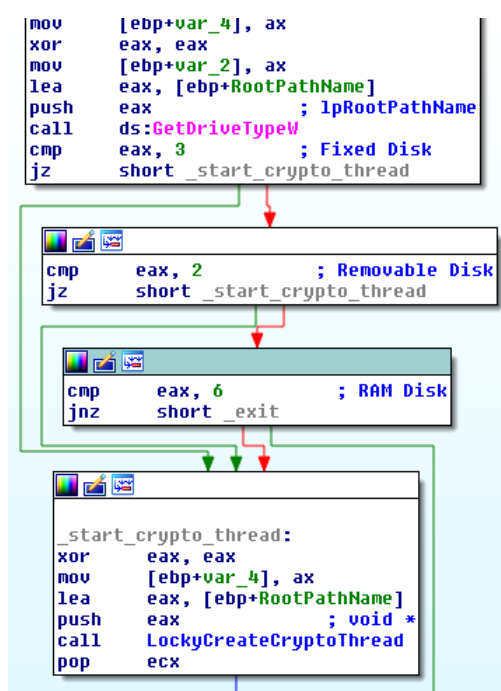


Ilustración 11. Algoritmo del cifrado del código dañino

El código dañino accede a todas las unidades de disco de tipo disco duro, discos extraíbles y discos RAM. Además de estas unidades de disco, también enumera y accede a los recursos de red, incluso si no están montados. El acceso a las unidades de red se gestiona mediante el uso de funciones de la librería "MPR", como "WNetEnumResource" y "WNetAddConnection2". La llamada a la función de búsqueda de recursos de red, se realiza de forma recursiva por cada recurso encontrado.

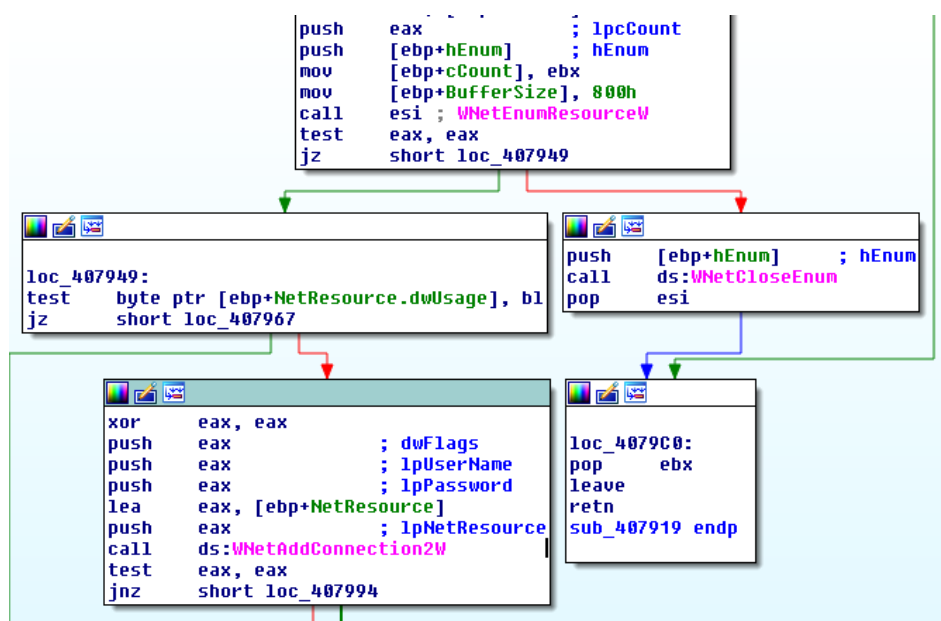


Ilustración 12. Búsqueda de recursos de red y su posterior conexión

Por cada carpeta y archivo encontrado, el código dañino comprueba que no tenga un nombre incluido en la siguiente lista y si así fuera, sería ignorado y no lo cifraría.

Windows	Program Files (x86)
Boot	AppData
System Volume Information	Application Data
\$Recycle.Bin	winnt
thumbs.db	tmp
temp	_Locky_recover_instructions.txt
Program Files	_Locky_recover_instructions.bmp

El código dañino guarda en memoria un gran listado de todos los archivos y de las rutas donde se encuentra en el sistema comprometido. Una vez que acaba de enumerar todos los archivos y carpetas, procede a cifrarlos uno por uno. Para ello lo primero que realiza es importar la clave pública RSA que obtuvo del servidor C2, mediante una llamada a *"CryptImportKey"*.

00401386	50	push	eax	
00401387	FF15 5C104100	call	[<ADVAPI32.CryptImportKey>]	ADVAPI32.CryptImportKey
0040138D	85C0	test	eax, eax	
0040138F	75 1E	jnz	short 004013AF	
00401391	FF15 38114100	call	[<KERNEL32.GetLastError>]	ntdll.RtlGetLastWin32Error
00401397	8945 FC	mov	[ebp-4], eax	
0040139A	68 504F4100	push	00414F50	
0040139F	8D45 F8	lea	eax, [ebp-8]	
004013A2	50	push	eax	

ds:[0041105C]=77DEA1F1 (ADVAPI32.CryptImportKey)

00982988	06	02	00	00	00	A4	00	00	52	53	41	31	00	08	00	00	■■■■-■-■.RSA1.■..	0098FCE
009829C8	01	00	00	00	AD	D0	86	1F	2B	8A	D0	7F	25	AF	5C	F8	■.■.-v■■■-■Dx% \o	0098FCE
009829D8	09	CF	82	35	40	2F	5F	B6	40	59	D0	6B	F8	F2	39		ÜI5M/ qHVik0ü39	0098FCE
009829E8	9B	2C	82	A7	8A	20	3E	46	96	AD	63	4F	86	81	C3	EE	■.8■ >F■-c00■E■	0098FCE
009829F8	0F	8C	D9	66	04	3F	6B	AF	09	85	D2	E0	00	52	B5	B0	■00f0?k'■0ü3.Ry0	0098FCF
00982A08	51	F4	2D	B9	7C	CE	DE	10	0F	88	AB	30	BE	1C	B6	EE	Qd-1 Ib000000000000	0098FCF
00982A18	35	EA	54	6C	16	87	87	49	99	85	20	F8	6C	4F	63	55	5ê1000-100 010cU	0098FCF
00982A28	35	4C	5D	F7	E5	DB	F6	7A	00	AD	5F	C0	38	00	10	3A	5LJ>3ü0ê. i8000	0098FCF
00982A38	56	EF	8A	40	ED	EE	6F	D5	EA	28	40	88	19	B5	2C	12	U000000000000000000	0098FD0
00982A48	56	A2	BB	BF	7B	EC	0F	14	03	AC	A8	3A	F6	F8	C0	B2	0x>2i0000-00000000	0098FD0
00982A58	B5	ED	1C	8D	10	57	C3	85	10	36	E8	5F	A3	8E	D5	30	ui000000000000000000	0098FD0
00982A68	81	CC	95	26	99	96	64	98	3C	F0	5C	83	A5	92	D3	53	■000000000000000000	0098FD0
00982A78	23	B6	A4	6E	D6	64	AF	24	94	9B	5A	E6	39	C3	AF	C7	■q0n0ü \$0020091 0	0098FD1
00982A88	F5	62	6D	E4	48	F3	DE	74	68	D0	5C	AF	DF	F5	63	E9	0b000000000000000000	0098FD1
00982A98	31	5F	8D	FD	70	A5	AB	D6	55	D7	A9	D2	A4	6F	6A		1 000000000000000000	0098FD1
00982AA8	86	E3	34	74	4F	DF	82	44	60	A6	75	66	70	D0	20	92	0000 00000000000000	0098FD1
00982AB8	8C	BF	49	78	CD	2A	84	FD	36	A9	FA	A0	11	8A	8D	2D	000000000000000000	0098FD2
00982AC8	39	AC	73	D5	00	F0	AD	BA	00	F0	AD	BA	00	F0	AD	BA	9-50.0-0.0-0.0-0	0098FD2
00982AD8	09	AD	08	AD	00	AD	08	AD	00	AD	08	AD	00	AD	08	AD	000000000000000000	0098FD2

Tras importar la clave procede a generar un nombre aleatorio para el archivo cifrado. Este nombre está compuesto por dos partes: 16 caracteres generados aleatoriamente mediante llamadas a "CryptGenRandom" y el ID único generado al

principio de la ejecución. A este nombre de archivo se le añade la extensión ".locky" para las cuatros primeras versiones y con la extensión ".zepto" para la versiones posteriores.

A continuación, el archivo es movido a su misma localización pero con el nuevo nombre. De esa forma se realiza una copia del archivo original y borra la referencia del nombre antiguo en un solo paso, utilizando el mismo punto del disco físico para evitar el uso de herramientas de recuperación de archivos.

El código dañino genera una clave AES con un valor obtenido aleatoriamente usando "CryptGenRandom".

8D85 6CFBFFFF	lea	eax,	[ebp-494]
50	push	eax	
8B45 08	mov	eax,	[ebp+8]
6A 10	push	10	
FF30	push	dword ptr [eax]	
FF15 4C104100	call	[<&ADVAPI32.CryptGenRandom>]	

Ilustración 14. Generación de clave AES

Cifra con la clave AES todo el contenido del archivo asegurándose que la clave usada es borrada de la memoria al finalizar. A continuación, se añade al principio de los datos cifrados la clave AES cifrada con la clave pública RSA.

En cada carpeta que haya encontrado y podido cifrar algún archivo, el código dañino deja un archivo de texto que varía dependiendo de la versión. Para más detalles, consultar el apartado [ARCHIVOS DE RESCATE](#) del presente informe.

Una vez finalizado todo el proceso de cifrado de archivos, el código dañino conecta con su servidor C2 y le envía de forma cifrada un paquete con la siguiente información:

- El ID único generado por el código dañino.
- El número de archivos cifrados correctamente.
- El número de archivos que fallaron al ser cifrados.

```

mov     [ebp+arg_4], eax
lea     eax, [ebp+var_0C]
push    offset aId_0      ; "id="
push    eax
mov     edi, offset dword_418D10
mov     byte ptr [ebp+var_4], 0Bh
call    sub_404BF9
push    offset aActStatsPath ; "&act=stats&path="
push    eax
lea     eax, [ebp+var_1C8]
mov     byte ptr [ebp+var_4], 0Ch
call    sub_404C70
mov     ecx, [ebp+arg_4] ; int
lea     edi, [ebp+var_158]
mov     byte ptr [ebp+var_4], 0Dh
call    sub_404CA1
push    offset aEncrypted ; "&encrypted="
push    eax
lea     eax, [ebp+var_200]
mov     byte ptr [ebp+var_4], 0Eh
call    sub_404C70
add     esp, 40h
mov     ecx, [ebp+var_1C] ; int
lea     edi, [ebp+var_120]
mov     byte ptr [ebp+var_4], 0Fh
call    sub_404CA1
push    offset aFailed ; "&failed="

```

Ilustración 15. Datos enviados tras el proceso de cifrado

Una vez finalizado todo el proceso de cifrado, en las primeras dos versiones, el código dañino crea la siguiente entrada en el registro:

```
[HKEY_CURRENT_USER\Software\Locky\completed] = <0x00000001>
```

9. PERSISTENCIA EN EL SISTEMA

El código dañino crea la siguiente entrada para asegurar su persistencia y es borrada al finalizar su ejecución:

```
[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run\locky] =
<%TEMP%\svchost.exe>
```

El código dañino crea otra entrada en el registro en las dos primeras versiones del código dañino que, pese a no usarse como persistencia, mantiene información importante del código dañino.

```
[HKEY_CURRENT_USER\Software\Locky]
```

En la tercera versión del código dañino se crea la entrada con un nombre aleatorio mientras que a partir de la cuarta versión no se crea ninguna entrada en el registro.

10.1 PRIMERA VERSIÓN DEL ALGORITMO DGA

A continuación se muestra la representación en código *python* del primer algoritmo.

```

max_bits=32
ROL4 = lambda val, r_bits: \
    (val << r_bits%max_bits) & (2**max_bits-1) | \
    ((val & (2**max_bits-1)) >> (max_bits-(r_bits%max_bits)))
ROR4 = lambda val, r_bits: \
    ((val & (2**max_bits-1)) >> r_bits%max_bits) | \
    (val << (max_bits-(r_bits%max_bits)) & (2**max_bits-1))
#day 1..31
#month 1..12
#year 1601..30827
def gen(year, month, day, idx):
    j = 0;
    v21 = 0;
    v3 = ROR4(0xB11924E1 * (year + 7157), 5);
    v4 = ROR4(0xB11924E1 * (v3 + (day >> 1) + 655360001), 5);
    v5 = ROR4(0xB11924E1 * (v4 + month + 654943060), 5);
    v6 = ROL4(idx % 6, 21);
    v7 = ROR4(0xB11924E1 * (v5 + v6 + 655360001), 5);
    v23 = (v7 + 655360001)%(2**32);
    name_size = (v7 + 655360001) % 0xB + 5;
    alloc_size = (v7 + 655360001) % 0xB + 8;
    domain = ''
    for idx in range(name_size):
        v9 = ROL4(v23, idx);
        v11 = ROR4(0xB11924E1 * v9, 5);
        v12 = (v11 + 655360001)%2**32;
        v23 = v12;
        domain += chr(v12 % 25 + ord('a'))
    domain += "."
    v15 = ROR4((0xB11924E1 * v23)%2**32, 5);
    v16 = ((v15 + 655360001)%2**32) % 0xE;
    domain += ['ru', 'pw', 'eu', 'in', 'yt', 'pm', 'us', 'fr', 'de', 'it', 'be', 'uk',
'n', 'tf'][v16]
    return domain
urls = []
#for year in range(2016,2017):
for year in range(2016,2017):
    for month in range(1,12):
        for day in range(1,31):
            for idx in range(6):
                urls += [gen(year, month, day, idx)]
print urls

```

10.2 SEGUNDA VERSIÓN DEL ALGORITMO DGA

A continuación se muestra la representación en código *python* del segundo algoritmo.

```

max_bits=32
ROL4 = lambda val, r_bits: \
    (val << r_bits%max_bits) & (2**max_bits-1) | \
    ((val & (2**max_bits-1)) >> (max_bits-(r_bits%max_bits)))
ROR4 = lambda val, r_bits: \
    ((val & (2**max_bits-1)) >> r_bits%max_bits) | \
    (val << (max_bits-(r_bits%max_bits)) & (2**max_bits-1))
def gen(year, month, day, idx, seed):
    j = 0;
    v21 = 0;
    v3 = ROR4(0xB11924E1 * (year + 7157), 7);
    v3 = ROR4(0xB11924E1 * (v3 + seed + 655360001), 7);
    v4 = ROR4(0xB11924E1 * (v3 + (day >> 1) + 655360001), 7);
    v5 = ROR4(0xB11924E1 * (v4 + month + 654943060), 7);
    seed = ROL4(seed, 17);
    v6 = ROL4(idx & 7, 21);
    v7 = ROR4(0xB11924E1 * (v5 + v6 + seed + 655360001), 7);
    v23 = (v7 + 655360001)%(2**32);
    name_size = v23 % 0xB + 5;
    alloc_size = v23 % 0xB + 8;
    domain = ''
    for idx in range(name_size):
        v9 = ROL4(v23, idx);
        v11 = ROR4(0xB11924E1 * v9, 7);
        v12 = (v11 + 655360001)%2**32;
        v23 = v12;
        domain += chr(v12 % 25 + ord('a'));
    domain += "."
    v15 = ROR4((0xB11924E1 * v23)%2**32, 7);
    v16 = ((v15 + 655360001)%2**32) % 0xE;
    domain
    ['ru','pw','eu','in','yt','pm','us','fr','de','it','be','uk','nl','tf'][v16] +=
    return domain
urls = []
year = 2016
month = 12
day = 24
seed = 7
for idx in range(8):
    urls += [gen(year, month, day, idx, seed)];
print "%4d-%2.2d-%2.2d | Seed= %s |: %s" % (year, month, day, seed, urls);

```

11. ARCHIVOS RELACIONADOS

Los archivos relacionados con el código dañino son los siguientes:

<%TEMP%>			
Nombre	Fecha Creación	Tamaño bytes	Hash SHA1
svchost.exe	<varía>	<varía>	<varía>
<ruta_variable>			
Nombre	Fecha Creación	Tamaño bytes	Hash SHA1
Locky_recover_instructions.txt	<varía>	<varía>	<varía>
Locky_recover_instructions.htm	<varía>	<varía>	<varía>
Locky_recover_instructions.bmp	<varía>	<varía>	<varía>
_HELP_instructions.txt (v.3)	<varía>	<varía>	<varía>
_HELP_instructions.bmp (v.3)	<varía>	<varía>	<varía>
reperr.log (v.7)	<varía>	70 bytes	<varía>
_<valor_número>_HELP_instructions.html(v.4/5)	<varía>	<varía>	<varía>
<%DESKTOP%>			
Nombre	Fecha Creación	Tamaño bytes	Hash SHA1
Locky_recover_instructions.htm	<varía>	<varía>	<varía>
Locky_recover_instructions.bmp	<varía>	<varía>	<varía>
_HELP_instructions.txt (v.3)	<varía>	<varía>	<varía>
_HELP_instructions.bmp (v.3/5)	<varía>	<varía>	<varía>
_HELP_instructions.html (v.5)	<varía>	<varía>	<varía>
_<valor_número>_HELP_instructions.txt (v.4/5)	<varía>	<varía>	<varía>
_<valor_número>_HELP_instructions.html(v.4/5)	<varía>	<varía>	<varía>

12. DETECCIÓN

Para detectar si un equipo se encuentra o ha estado infectado para cualquiera de sus usuarios, se emplearán alguna de las herramientas de Mandiant, como el "Mandiant IOC Finder" o el colector generado por RedLine, con los indicadores de compromiso generados para su detección.

Aparte de las herramientas citadas anteriormente, se podrán usar herramientas del sistema como el Editor de Registro del sistema.

12.1 HERRAMIENTAS DEL SISTEMA

Una vez arrancado el Editor del Registro (*Inicio -> Ejecutar -> regedit.exe*) se buscará la rama:

[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run\locky]	=
<%TEMP%\svchost.exe>	

En el caso de que dicha entrada exista, es un claro compromiso del sistema. El que la entrada exista implica que el código dañino ha fallado en su intento de cifrar los archivos o se encuentra en proceso de esta acción.

También se deberá buscar la siguiente entrada:

[HKEY_CURRENT_USER\Software\Locky]

Si la entrada existe, es un claro indicador del compromiso del sistema por parte del código dañino. Si el código dañino se ejecutó con éxito y acabó el proceso de cifrado se podrá localizar la siguiente clave:

[HKEY_CURRENT_USER\Software\Locky\completed] = <0x00000001>

En el caso de la tercera versión del código dañino, se deberá buscar la siguiente entrada de registro con tres entradas, con nombres aleatorios e información codificada.

[HKEY_CURRENT_USER\Software\<valor_aleatorio>]

Otra evidencia clara del compromiso, es la desaparición de los archivos originales y la aparición de archivos con nombres aleatorios con la extensión ".locky" o ".zepto".

Aparte de estos archivos, la presencia de ciertos ficheros en numerosas carpetas implica un compromiso del sistema. El nombre de los archivos con la información del secuestro puede variar dependiendo de la versión del código dañino, pudiendo estar en la siguiente lista.

_Locky_recover_instructions.txt
_Locky_recover_instructions.html
_Locky_recover_instructions.bmp
_HELP_instructions.txt
_HELP_instructions.bmp
_<valor_númeroico_tres_digitos>_HELP_instructions.html
_HELP_instructions.html
_HELP_instructions.bmp

Por último, la presencia en el escritorio de un archivo HTML con el mismo nombre que los archivos de texto y el fondo de pantalla cambiado con la nota del secuestro de los archivos y procedimientos a seguir, son evidencias del compromiso del sistema.

La existencia de un archivo llamado "reperr.log" en algún directorio del sistema comprometido es un posible indicador de compromiso de la séptima versión del código dañino.

12.2 MANDIANT

Se ha generado un nuevo archivo indicador de compromiso. El nombre del indicador generado es con GUID "95645a49-9451-44bd-885b-f4e02b8e646e".

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

Una vez finalizó su ejecución con éxito, el código dañino no deja el ejecutable en el sistema comprometido, ni entrada de persistencia para asegurar su futura ejecución dado que su labor ha concluido y la extorsión ha sido lanzada.

Para intentar una desinfección previa a que finalice su ejecución se pueden realizar los siguientes pasos.

1. Con el Editor del Registro de Windows (Inicio -> Ejecutar -> regedit.exe) se buscará la siguiente clave:

[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run\locky] =
<%TEMP%\svchost.exe>

2. En caso de que dicha clave exista, se procederá a su borrado.
3. Se procederá a ejecutar el Gestor de Tareas de Windows (Inicio -> Ejecutar -> taskmgr) y se buscará un proceso llamado "svchost.exe" que no pertenezca al usuario SYSTEM y se procederá a parar su ejecución.
4. Usando el Explorador de Windows o equivalente, se irá a la carpeta %TEMP% del sistema comprometido y, si existe, se borrará el archivo llamado "svchost.exe".
5. Posteriormente se irá a la siguiente entrada del registro y si existe, se borrará.

[HKEY_CURRENT_USER\Software\Locky]

Si no existiese, se buscará la siguiente entrada del registro con tres valores de nombres aleatorios e información codificada, que se borrará.

[HKEY_CURRENT_USER\Software\<valor_aleatorio>]

6. Por último se deberán borrar todos los archivos de texto llamados "_Locky_recover_instructions.txt" que se encuentren en el sistema al igual que los dos archivos creados en el escritorio con el mismo nombre, pero con extensiones HTML y BMP. Tras ello se cambiará el fondo del escritorio al fondo deseado por el usuario.

En el caso de la tercera versión del código dañino, los dos archivos en el escritorio poseen otro nombre: "_HELP_instructions.<extensión>".

En el caso de que exista un archivo llamado "reperr.log" en el sistema comprometido que no pueda estar relacionado con otro programa o usuario se procederá a borrar.

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. VACUNA CONTRA EL CÓDIGO DAÑINO

En las dos primeras versiones del código dañino, se presentan una serie de fallos de diseño que pueden ser aprovechados por los usuarios o administradores de sistemas para evitar la carga del código dañino. Actualmente no se conoce otra vacuna para la tercera versión y posteriores del código dañino.

14.1 CHEQUEO DEL IDIOMA DEL SISTEMA

El código dañino procede en su arranque a comprobar el idioma que tiene el usuario activo y del sistema operativo en sí. Si el idioma es ruso procederá a eliminarse del sistema sin realizar su carga dañina.

Esta medida puede no ser adecuada para un gran número de usuarios ya que cambiaría el teclado a cirílico. Pese a ello es la medida más simple de realizar, debido a que no requiere de conocimientos técnicos.

14.2 ENTRADA "COMPLETED" E "ID" EN EL REGISTRO

El código dañino tras finalizar su carga dañina de forma completa y para evitar múltiples cifrados en el mismo sistema, crea una entrada llamada "completed" con un valor a "1" en su rama de registro:

`[HKEY_CURRENT_USER\Software\Locky]`

En el arranque del código dañino, se comprueba la existencia de dicha entrada "completed" y en el caso de que se encuentre con un valor "1", se obtiene el valor de la entrada "Id" y se procede a recalcular el identificador del sistema. El algoritmo utilizado para la creación del identificador es explicado de forma detallada en el punto [CARACTERÍSTICAS TÉCNICAS](#) del presente informe.

Tras el cálculo, se compara el valor introducido en el registro con el calculado y, si coinciden, el código dañino procede a borrarse a sí mismo y no realizar su carga dañina.

Teniendo en cuenta esto, puede crearse la entrada de registro principal indicada en el recuadro superior y crear los valores "completed" a "1" e "Id", con el id ya pre-calculado para el sistema. De esta forma si el código dañino se ejecuta en el sistema se impedirá que cifre los archivos.

14.3 MODIFICACIÓN DE LOS PERMISOS DE ACCESO AL REGISTRO

El código dañino, en su arranque, intenta acceder a la entrada de registro principal:

[HKEY_CURRENT_USER\Software\Locky]

En el caso de que ésta exista y pueda acceder, continúa su ejecución, y en el caso de que no exista, la crea. Sin embargo, si no puede acceder a la entrada el código dañino procede a eliminarse a sí mismo y finalizar su ejecución sin ejecutar su carga dañina.

Es por ello, que se puede crear la siguiente entrada en el registro y tras haberla creado, proceder a eliminar todos los accesos a ella para cualquier usuario modificando las ACL ("Access Control List").

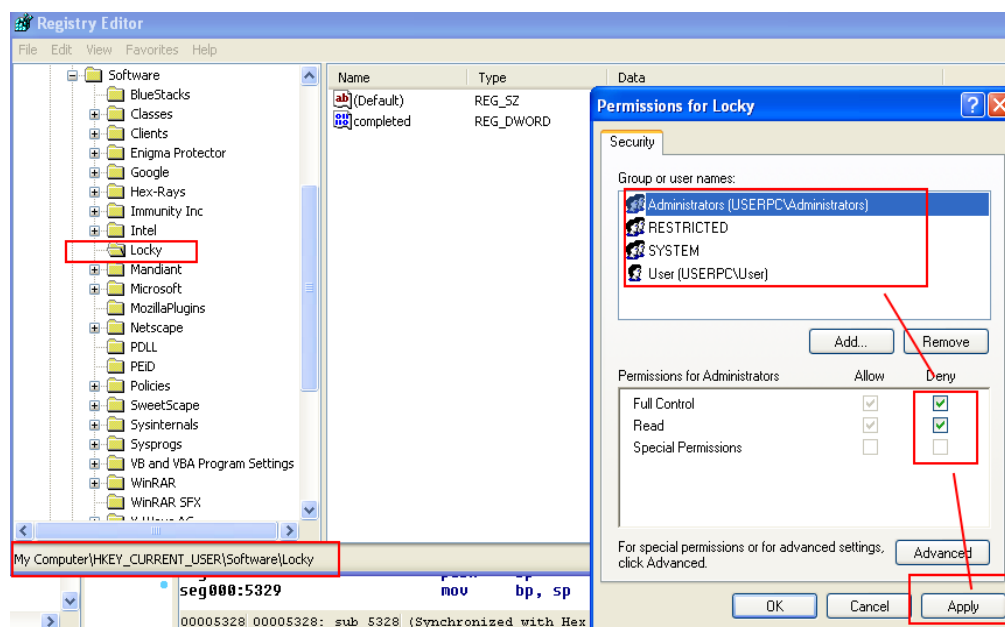


Ilustración 17. Modificación de los permisos de acceso a la entrada del registro

14.4 CREACIÓN DE CLAVE RSA PÚBLICA CORRUPTA

El código dañino guarda en su entrada principal en el registro la clave RSA pública obtenida del servidor C2, para cifrar la clave AES usada en el cifrado de los archivos del sistema comprometido. La clave es guardada en la entrada de registro llamada "pubkey".

Si dicha entrada existe y la clave puede importarse, el código dañino usará esa clave en lugar de descargar una del C2. Sin embargo, si se falla al importar dicha clave el código dañino procede a eliminarse a sí mismo y no cifra ningún archivo.

Es por ello, que si se crea esa entrada con un valor de tipo *BINARY* con un solo byte (da igual su contenido), la importación de la clave sería errónea y se produciría el efecto deseado.

14.5 CREACIÓN DE CLAVE RSA PÚBLICA CON VALOR PRE-CALCULADO

Tal y como se indicó en el punto anterior, el código dañino busca en el registro una entrada llamada "pubkey", dentro de su clave principal. Si la encuentra y puede importar la clave obtenida de ella, no descargará del servidor C2 una nueva clave.

Es por ello que si se crean un par de claves RSA (una pública para cifrar y otra privada para descifrar) y se introduce la pública en el registro en dicha entrada, el código dañino usaría la clave pre computada para cifrar los archivos.

Una vez realizado todo el proceso de cifrado, y al tener la clave privada para descifrar los archivos, se podrían recuperar sin problemas y sin tener que pagar ningún tipo de rescate.

15. INFORMACIÓN DEL ATACANTE

La muestra analizada del código dañino conecta a una serie de direcciones IP embebidas en su código.

37.235.53.18
91.195.12.131
149.154.157.14
151.236.14.51
78.40.108.39
51.254.240.48
91.219.29.41
185.82.216.55
217.12.223.83

15.1 78.40.108.39

La información de WHOIS de la dirección IP "78.40.108.39" es la siguiente:

```

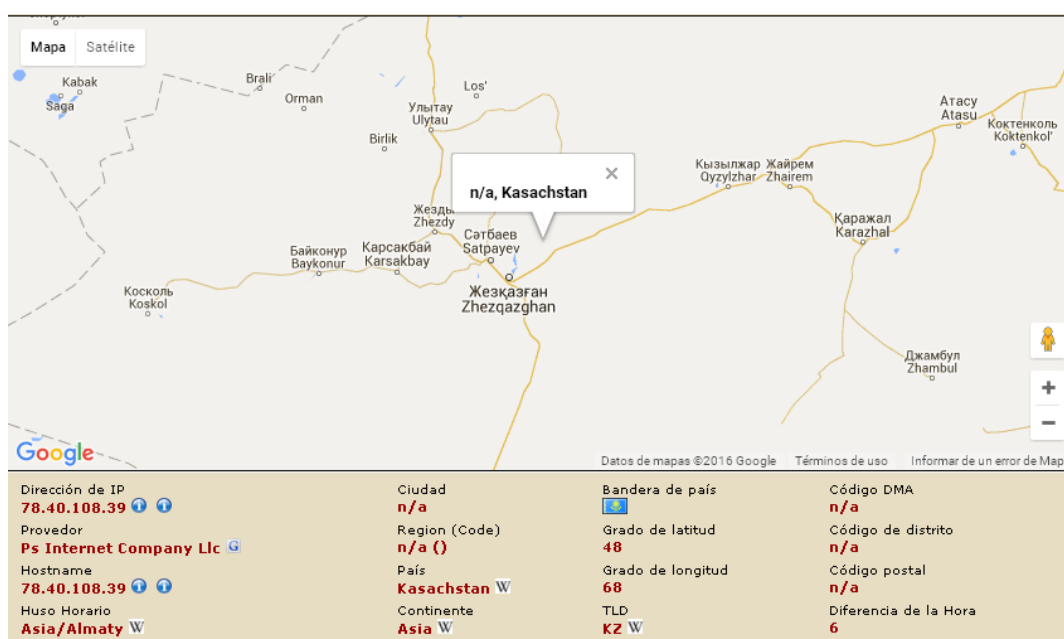
inetnum:      78.40.108.0 - 78.40.108.127
netname:      PS-KVM-3
descr:        78.40.108.0/25
country:      KZ
admin-c:      NOC-PS
tech-c:       NOC-PS
status:       ASSIGNED PA
mnt-by:       MNT-PS
mnt-routes:   MNT-PS
mnt-domains:  MNT-PS
created:      2015-08-05T07:29:14Z
last-modified: 2015-08-18T08:48:21Z
source:       RIPE
  
```

```

role:          NOC PS
address:       PS Internet Company LLC
address:       117A Makataev street, office 201
address:       050000 Almaty Kazakhstan
phone:         +7 727 388 82 31
admin-c:       NIK7-RIPE
admin-c:       KP2832-RIPE
tech-c:        KP2832-RIPE
tech-c:        NIK7-RIPE
nic-hdl:       NOC-PS
  
```

Ilustración 18. Información WHOIS de la dirección IP 78.40.108.39

15.1.1 GEOLOCALIZACIÓN



Dirección de IP 78.40.108.39	Ciudad n/a	Bandera de país 	Código DMA n/a
Proveedor Ps Internet Company Llc	Region (Code) n/a ()	Grado de latitud 48	Código de distrito n/a
Hostname 78.40.108.39	País Kasachstan	Grado de longitud 68	Código postal n/a
Huso Horario Asia/Almaty	Continente Asia	TLD KZ	Diferencia de la Hora 6

Ilustración 19. Geolocalización de la dirección IP 78.40.108.39

15.2 91.195.12.131

La información de WHOIS de la dirección IP "91.195.12.131" es la siguiente:

Db Range	91.195.12.0 - 91.195.13.255
Published Range	91.195.12.0 - 91.195.13.255
IP Location	 Ukraine Kiev Pe Astakhov Pavel Viktorovich
ASN	 AS57492 HOST4BIZ-NET PE Astakhov Pavel Viktorovich (registered Nov 10, 2011)
Resolve Host	91-195-12-131.net.host4.biz
Whois Server	whois.ripe.net
IP Address	91.195.12.131

Ilustración 20. Información WHOIS de la dirección IP 91.195.12.131

15.2.1 GEOLOCALIZACIÓN

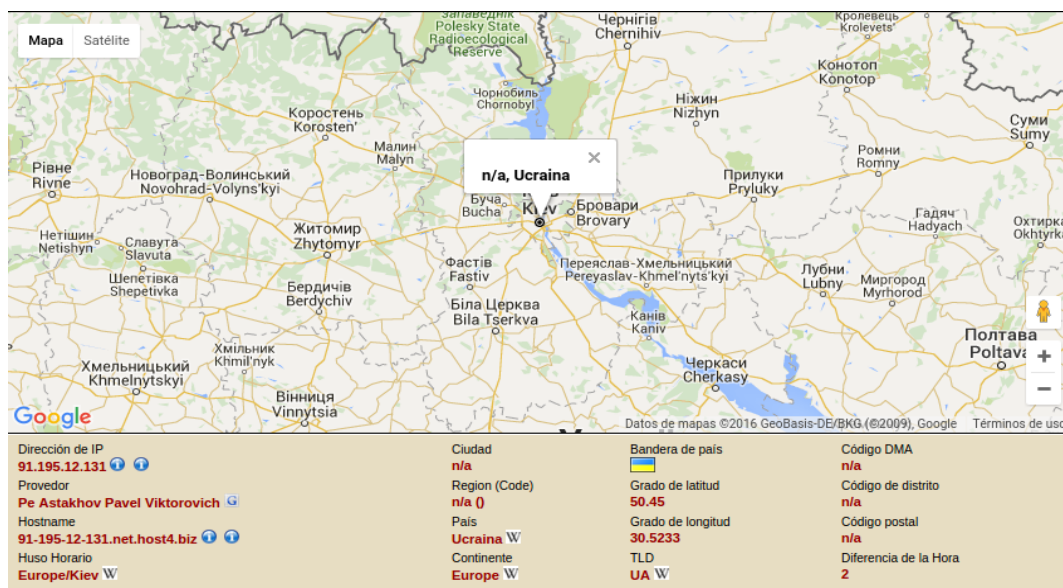


Ilustración 21. Geolocalización de la dirección IP 91.195.12.131

15.3 149.154.157.14

La información de WHOIS de la dirección IP "149.154.157.14" es la siguiente:

Db Range	149.154.157.0 - 149.154.157.255
Published Range	149.154.157.0 - 149.154.157.255
IP Location	 Italy Milan Edis GmbH
ASN	 AS20836 CDLAN-AS CDLAN s.r.l. (registered Jun 12, 2001)
Resolve Host	339.it.multiservers.xyz
Whois Server	whois.ripe.net
IP Address	149.154.157.14

Ilustración 22. Información WHOIS de la dirección IP 149.154.157.14

15.3.1 GEOLOCALIZACIÓN



Ilustración 23. Geolocalización de la dirección IP 149.154.157.14

15.4 151.236.14.51

La información de WHOIS de la dirección IP "151.236.14.51" es la siguiente:

Db Range	151.236.14.0 - 151.236.14.255
Published Range	151.236.14.0 - 151.236.14.255
IP Location	 Netherlands Amsterdam Edis GmbH
ASN	 AS43350 NFORCE NForce Entertainment BV (registered Jul 18, 2007)
Resolve Host	339.nl.multiservers.xyz
Whois Server	whois.ripe.net
IP Address	151.236.14.51

Ilustración 24. Información WHOIS de la dirección IP 151.236.14.51

15.4.1 GEOLOCALIZACIÓN



Ilustración 25. Geolocalización de la dirección IP 151.236.14.51

15.5 37.235.53.18

La información de WHOIS de la dirección IP "37.235.53.18" es la siguiente:

Db Range	37.235.53.0 - 37.235.53.255
Published Range	37.235.53.0 - 37.235.53.255
IP Location	 Spain Madrid Edis GmbH
ASN	 AS39020 COMVIVE-AS Comvive Servidores S.L. (registered Nov 29, 2005)
Resolve Host	339.es.multiservers.xyz
Whois Server	whois.ripe.net
IP Address	37.235.53.18

Ilustración 26. Información WHOIS de la dirección IP 37.235.53.18

15.5.1 GEOLOCALIZACIÓN



Ilustración 27. Geolocalización de la dirección IP 37.235.53.18

15.6 51.254.240.48

La información de WHOIS de la dirección IP "51.254.240.48" es la siguiente:

IP Location	 France Roubaix Relink Llc Andrey Orlov
ASN	 AS16276 OVH , FR (registered Feb 15, 2001)
Whois Server	whois.ripe.net
IP Address	51.254.240.48

% Abuse contact for '51.254.240.0 - 51.254.240.255' is 'abuse@ovh.net'	
inetnum:	51.254.240.0 - 51.254.240.255
netname:	OVH_93228240
descr:	OVH Static IP
country:	FR
org:	ORG-RLA03-RIPE
admin-c:	OTC2-RIPE
tech-c:	OTC2-RIPE
status:	LEGACY
mnt-by:	OVH-MNT
created:	2015-10-29T10:46:06Z
last-modified:	2015-10-29T10:46:06Z
source:	RIPE
organisation:	ORG-RLA03-RIPE
org-name:	Relink LLC Andrey Orlov
org-type:	OTHER

Ilustración 28. Información WHOIS de la dirección IP 51.254.240.48

15.6.1 GEOLOCALIZACIÓN



Ilustración 29. Geolocalización de la dirección IP 51.254.240.48

15.7 91.219.29.41

La información de WHOIS de la dirección IP "91.219.29.41" es la siguiente:

IP Location	 Ukraine Kiev Flp Kochenov Aleksey Vladislavovich
ASN	 AS3254 LUCKYNET Lucky Net Ltd, UA (registered Sep 19, 1994)
Resolve Host	41.29.219.91.colo.ukrserver.com
Whois Server	whois.ripe.net
IP Address	91.219.29.41


```

% Abuse contact for '91.219.28.0 - 91.219.31.255' is 'hostmaster@uadomen.com'

inetnum:          91.219.28.0 - 91.219.31.255
netname:          UKRSERVERS-NET
country:          UA
org:              ORG-FKAV1-RIPE
admin-c:          KCH78-RIPE
tech-c:           KCH78-RIPE
status:           ASSIGNED PI
mnt-by:           RIPE-NCC-END-MNT
mnt-by:           UADOMEN-MNT
mnt-routes:       UADOMEN-MNT
mnt-routes:       DATAHARBOUR-MNT
mnt-domains:      UADOMEN-MNT
created:          2010-09-06T11:31:55Z
last-modified:    2016-04-14T11:03:38Z
source:           RIPE
sponsoring-org:   ORG-VI4-RIPE
  
```

Ilustración 30. Información WHOIS de la dirección IP 91.219.29.41

15.7.1 GEOLOCALIZACIÓN

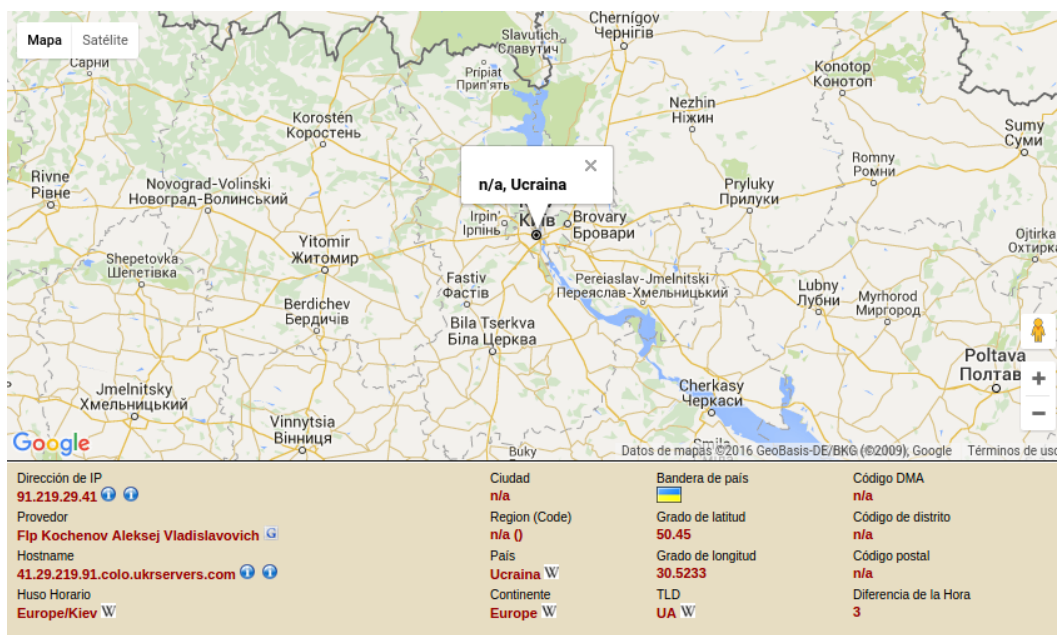


Ilustración 31. Geolocalización de la dirección IP 91.219.29.41

15.8 185.82.216.55

La información de WHOIS de la dirección IP "185.82.216.55" es la siguiente:

IP Location	 Bulgaria Sofia Itl-bulgaria Ltd.
ASN	 AS59729 ITL-BG , BG (registered Oct 27, 2014)
Resolve Host	bahromkina2.example.com
Whois Server	whois.ripe.net
IP Address	185.82.216.55


```

% Abuse contact for '185.82.216.0 - 185.82.217.255' is 'abuse@itldc.com'

inetnum:        185.82.216.0 - 185.82.217.255
netname:        ITLDC1-SOF1
descr:          ITL Sofia Datacenter Network
country:        BG
admin-c:        DD7045-RIPE
tech-c:         DD7045-RIPE
status:         ASSIGNED PA
mnt-by:         ITLBG-MNT
mnt-lower:      ITLBG-MNT
mnt-routes:     ITLBG-MNT
mnt-routes:     ITLBG-MNT
created:        2015-01-02T11:54:54Z
last-modified:  2015-01-02T17:30:32Z
source:         RIPE

person:         Dmitry Deineka
address:        Zornitsa district Suite 1 Nessebar 8240, BG
phone:          +359877639914
e-mail:         dmitry@itldc.com
nic-hdl:        DD7045-RIPE
mnt-by:         ITLBG-MNT
  
```

Ilustración 32. Información WHOIS de la dirección IP 185.82.216.55

15.8.1 GEOLOCALIZACIÓN

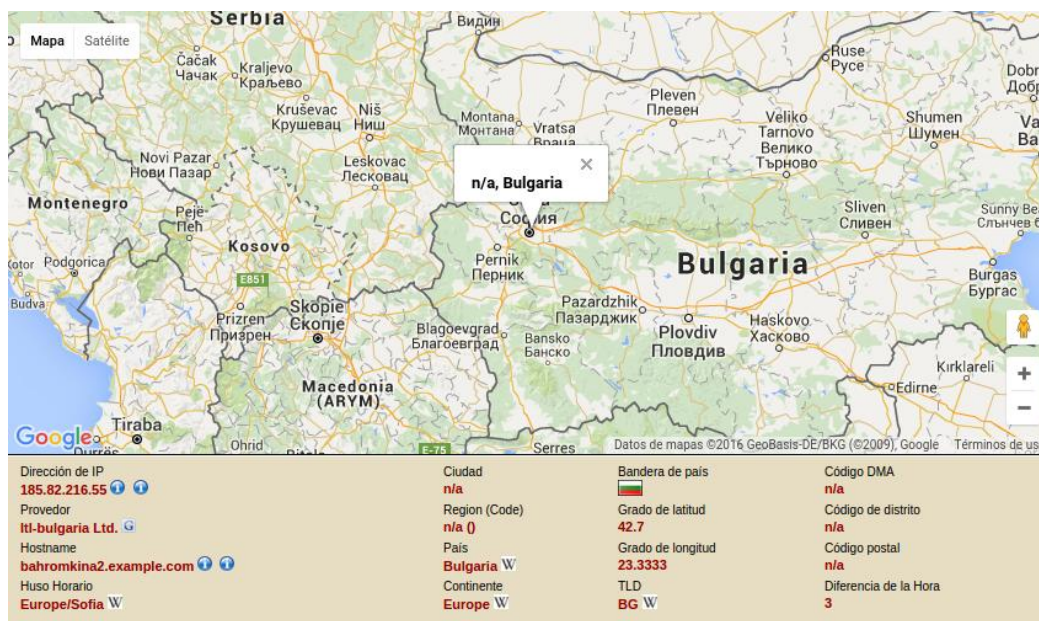


Ilustración 33. Geolocalización de la dirección IP 185.82.216.55

15.9 217.12.223.83

La información de WHOIS de la dirección IP "217.12.223.83" es la siguiente:

IP Location	 Ukraine Kharkiv Itl Company
ASN	 AS15626 ITLAS , UA (registered Aug 28, 2000)
Resolve Host	bahromkina.example.com
Whois Server	whois.ripe.net
IP Address	217.12.223.83


```

% Abuse contact for '217.12.223.0 - 217.12.223.127' is 'abuse@uaservers.net'

inetnum:        217.12.223.0 - 217.12.223.127
netname:        ADSYSTEMS
descr:          Kharkov
country:        UA
admin-c:        DD518-RIPE
tech-c:         DD518-RIPE
status:         ASSIGNED PA
notify:         axl@itl.ua
mnt-by:         ITL-MNT
created:        2003-06-20T10:05:41Z
last-modified:  2004-03-29T14:48:52Z
source:         RIPE

person:         Dmitry Deineka
remarks:        Please use abuse mailbox for any reports about network abuse. Thank
you.
address:        Zornitsa district Suite 1
address:        Nessebar, Burgas reg.
address:        Bulgaria 8240
mnt-by:         DEINEKA-MNT
phone:          +359877639914
  
```

Ilustración 34. Información WHOIS de la dirección IP 217.12.223.83

15.9.1 GEOLOCALIZACIÓN

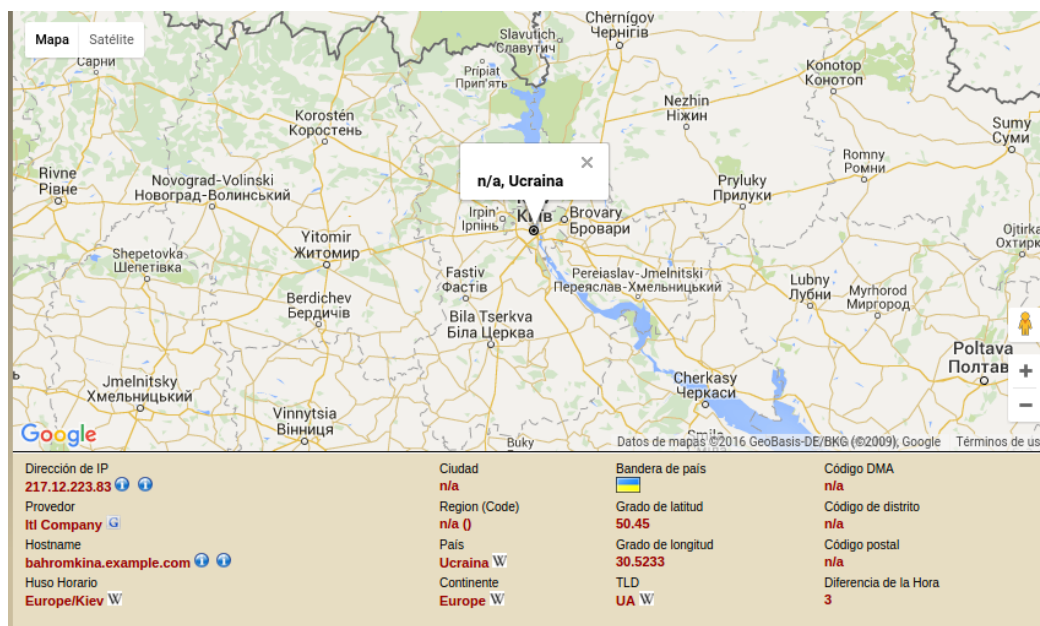


Ilustración 35. Geolocalización de la dirección IP 217.12.223.83

16. REFERENCIAS

<https://blogs.mcafee.com/mcafee-labs/locky-ransomware-rampage-JavaScript-downloader/>

<https://blog.avast.com/a-closer-look-at-the-locky-ransomware>

<https://blog.malwarebytes.org/intelligence/2016/03/look-into-locky/>

<http://blog.fortinet.com/post/a-closer-look-at-locky-ransomware-2>

<http://community.hpe.com/t5/Security-Research/Feeling-even-Locky-er/ba-p/6834311#.VufAsdf2Okr>

<http://www.redeszone.net/2016/03/11/el-troyano-bancario-dridex-deja-su-sitio-a-locky-en-la-botnet/>

17. REGLAS DE DETECCIÓN

17.1 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="95645a49-9451-44bd-885b-f4e02b8e646e"
  last-modified="2016-05-10T07:44:28" xmlns="http://schemas.mandiant.com/2010/ioc">
  <short_description>RANSOM.LOCKY</short_description>
  <description>IOC para detectar RANSOM.LOCKY</description>
  <authored_by>CCN-CERT</authored_by>
  <authored_date>2016-03-15T07:26:16</authored_date>
  <links />
  <definition>
    <Indicator operator="OR" id="244529fb-3424-43da-a3b6-e3d7815d2465">
      <IndicatorItem id="9fc7eaec-c925-43a6-9af5-272b7bd8f037" condition="is">
        <Context document="RegistryItem" search="RegistryItem/Path" type="mir" />
        <Content
type="string">HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run</Content>
      </IndicatorItem>
      <IndicatorItem id="53270c8e-cf22-4e8b-81c0-515fa67b9f8b" condition="is">
        <Context document="RegistryItem" search="RegistryItem/Path" type="mir" />
        <Content type="string">HKEY_CURRENT_USER\SOFTWARE\Locky</Content>
      </IndicatorItem>
      <Indicator operator="AND" id="bd24e398-d328-4898-bae4-f9f0caa1434c">
        <IndicatorItem id="f3694108-3005-44ce-89e1-f1abf9cbe49a" condition="is">
          <Context document="RegistryItem" search="RegistryItem/ValueName" type="mir" />
          <Content type="string">Locky</Content>
        </IndicatorItem>
        <Indicator operator="AND" id="022ec309-79cb-4f7f-a694-eab093df2d99">
          <IndicatorItem id="19aa7645-e69a-4639-9d2f-d31162746a4d" condition="contains">
            <Context document="RegistryItem" search="RegistryItem/Text" type="mir" />
            <Content type="string">svchost.exe</Content>
          </IndicatorItem>
        </Indicator>
      <Indicator operator="OR" id="3e582148-19a9-40ba-8745-7f4db875be11">
        <IndicatorItem id="bb940336-165f-45d2-8609-4243056c456d" condition="is">
          <Context document="RegistryItem" search="RegistryItem/ValueName" type="mir" />
          <Content type="string">opt321</Content>
        </IndicatorItem>
        <Indicator operator="AND" id="1ace7979-610f-4de6-b7a8-220e6cb97c61">
          <IndicatorItem id="0c7c0a12-c32c-445d-963c-022c031e326c" condition="contains">
            <Context document="RegistryItem" search="RegistryItem/Value" type="mir" />
            <Content type="string">svchost.exe</Content>
          </IndicatorItem>
        </Indicator>
      </Indicator>
    </Indicator>
    <Indicator operator="AND" id="718ae2f2-d8be-4627-b7f4-c944365524ad">
      <IndicatorItem id="cf9678bf-40c2-441e-a4b1-48daae2b7b30" condition="is">
        <Context document="RegistryItem" search="RegistryItem/ValueName" type="mir" />
        <Content type="string">id</Content>
      </IndicatorItem>
    </Indicator>
  </definition>
</ioc>
```

```

</IndicatorItem>
<Indicator operator="OR" id="b4d86371-534d-47e2-a673-6fb730f7d497">
  <IndicatorItem id="1a5b66ac-2e0a-4352-aa9c-28647af41ae3" condition="is">
    <Context document="RegistryItem" search="RegistryItem/ValueName" type="mir" />
    <Content type="string">paytext</Content>
  </IndicatorItem>
</Indicator>
<Indicator operator="OR" id="0defcec0-6310-4873-9e7e-e044fa62e28a">
  <IndicatorItem id="6de725ae-d630-4640-8b5f-db0b4afa24b6" condition="is">
    <Context document="RegistryItem" search="RegistryItem/Path" type="mir" />
    <Content type="string">pubkey</Content>
  </IndicatorItem>
</Indicator>
<Indicator operator="OR" id="74806707-e0b8-4f0d-8f7a-e01d7dde8ae5">
  <IndicatorItem id="f17f46b1-fb52-4875-9209-67816ead87b9" condition="is">
    <Context document="RegistryItem" search="RegistryItem/ValueName" type="mir" />
    <Content type="string">completed</Content>
  </IndicatorItem>
</Indicator>
</Indicator>
</definition>
</ioc>

```

17.2 YARA

```

rule Locky {
meta:
  description = "Regla para detectar RANSOM.LOCKY"
  author = "CCN-CERT"
  version = "1.0"
strings:
  $ = { 2E 00 6C 00 6F 00 63 00 6B 00 79 00 00 }
  $ = { 00 5F 00 4C 00 6F 00 63 00 6B 00 79 00 }
  $ = { 5F 00 72 00 65 00 63 00 6F 00 76 00 65 }
  $ = { 00 72 00 5F 00 69 00 6E 00 73 00 74 00 }
  $ = { 72 00 75 00 63 00 74 00 69 00 6F 00 6E }
  $ = { 00 73 00 2E 00 74 00 78 00 74 00 00 }
  $ = { 53 6F 66 74 77 61 72 65 5C 4C 6F 63 6B 79 00 }
condition:
  all of them
}

```