Informe Código Dañino
CCN-CERT ID-02/20

LokiBot

Febrero 2020

Edita:

GOBIERNO DE ESPAÑA MINISTERIO DE DEFENSA

CCN centro criptológico nacional

Fecha de Edición: febrero de 2020

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

ÍNDICE

1. SOBRE CCN-CERT, CERT GUBERNAMENTAL NACIONAL

El CCN-CERT es la Capacidad de Respuesta a incidentes de Seguridad de la Información del Centro Criptológico Nacional, CCN, adscrito al Centro Nacional de Inteligencia, CNI. 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 CNI, el RD 421/2004 de regulación del CCN y en el RD 3/2010, de 8 de enero, regulador del Esquema Nacional de Seguridad (ENS), modificado por el RD 951/2015 de 23 de octubre.

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, incluyendo la coordinación a nivel público estatal de las distintas Capacidades de Respuesta a Incidentes o Centros de Operaciones de Ciberseguridad existentes.

Todo ello, con el fin último de conseguir un ciberespacio más seguro y confiable, preservando la información clasificada (tal y como recoge el art. 4. F de la Ley 11/2002) y la información sensible, defendiendo el Patrimonio Tecnológico español, formando al personal experto, aplicando políticas y procedimientos de seguridad y empleando y desarrollando las tecnologías más adecuadas a este fin.

De acuerdo a esta normativa y la Ley 40/2015 de Régimen Jurídico del Sector Público es competencia del CCN-CERT la gestión de ciberincidentes que afecten a cualquier organismo o empresa pública. En el caso de operadores críticos del sector público la gestión de ciberincidentes se realizará por el CCN-CERT en coordinación con el CNPIC.

2. RESUMEN EJECUTIVO

El presente informe recoge el análisis del binario identificado por la firma SHA256 ff3e61a10ae528487ed74feb73108a2e473d4582c5908f66643fdf002dc67e4c, perteneciente a la familia del *infostealer* **LokiBot**.

Los orígenes de LokiBot se remontan a 2015 y su autoría se atribuye al actor **Carter/Lokistov**, que además de ser el desarrollador, se encargaba de su comercialización en diversos foros. El principal objetivo de LokiBot es la **exfiltración de información sensible** del sistema infectado y dada su naturaleza, también es **capaz de descargar muestras adicionales de malware**.

La principal vía de distribución que se suele observar detrás de las infecciones de LokiBot, son campañas de *malspam* adjuntando ficheros dañinos que desencadenarán la posterior descarga del *infostealer*.

Como cualquier otro tipo de software, el malware no se encuentra exento de ser *crackeado* y por este mismo motivo, LokiBot es tan popular incluso cinco años después de su lanzamiento. A finales de 2016 se liberaron al dominio público las herramientas necesarias para generar muestras *crackeadas*, que junto con la filtración del código fuente del panel de control, desencadenó que LokiBot pasara de un modelo basado en *Malware as a Service* a *commodity* malware.

En los puntos posteriores se entra en detalles técnicos sobre el proceso de infección y capacidades de la muestra de malware analizada, así como de detalles de interés relativos al binario.

3. DETALLES GENERALES

El binario objeto de análisis se corresponde con un Portable Ejecutable (PE) de 32-bits desarrollado bajo el *framework* .NET e identificado por la siguiente firma SHA256.

Fichero	SHA256
info.exe	ff3e61a10ae528487ed74feb73108a2e473d4582c5908f66643fdf002dc67e4c

La primera referencia que se observa del binario data del 3 de septiembre de 2019. En las detecciones que realizan los anti-virus, se pueden apreciar las firmas **MSIL/Kryptic**. Se trata de firmas genéricas para *packers* desarrollados en .NET, mientras que los motores son capaces de detectar el *Microsoft Intermediate Language* (MSIL), las capas de ofuscación, cifrado o compresión del *packer* impiden que el motor pueda correr sus reglas de detección sobre el payload final.

Tras un proceso manual de *unpacking* se puede obtener el payload final, una muestra *crackeada* de **LokiBot** en su versión 1.8, *infostealer* desarrollado en C++ cuyo origen se remonta a 2015, que se identifica por la siguiente firma SHA256.

Fichero	SHA256
payload.exe	bebecfca484a2d8f5fad1ef57047c9d93e12aeed8ae64406c535586d82c169da

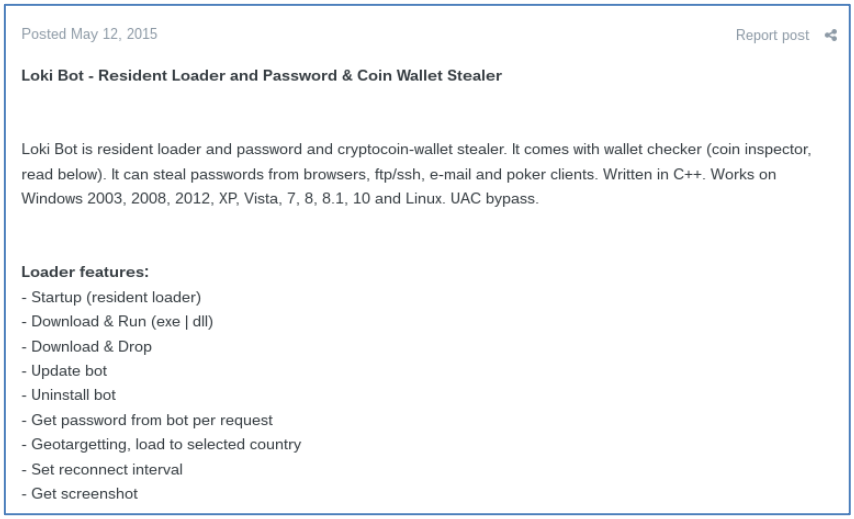
Las fechas de compilación de los binarios mencionados no aportan información útil a la investigación. Mientras que el *packer* ha modificado su fecha con el objetivo de prevenir que se pueda situar la infección en una línea temporal, el payload por su parte muestra la fecha de compilación del fichero original, del que se han servido posteriores *builders* para generar nuevas muestras *crackeadas*, como se detallará en puntos posteriores del informe.

Fichero	Fecha de compilación
info.exe	2 de abril de 1998 a las 16:50:51 UTC
payload.exe	23 de junio de 2016 a las 16:04:21 UTC

Cabe destacar que el código dañino no implementa técnicas de anti-análisis o detección de entornos virtualizados. Sin embargo, para hacer uso de las funciones de Windows, en lugar de importarlas en su *Import Address Table* (IAT) de forma directa, las resuelve de forma dinámica, identificando cada función a resolver a través de un hash precalculado para cada una de ellas.

4. MALWARE AS A SERVICE

El desarrollador de LokiBot, cuya actividad se encuentra registrada bajo los nombres de usuario **Lokistov** y **Carter**, entre otros, lanzó en 2015 la primera versión del código dañino.



Posted May 12, 2015 Report post

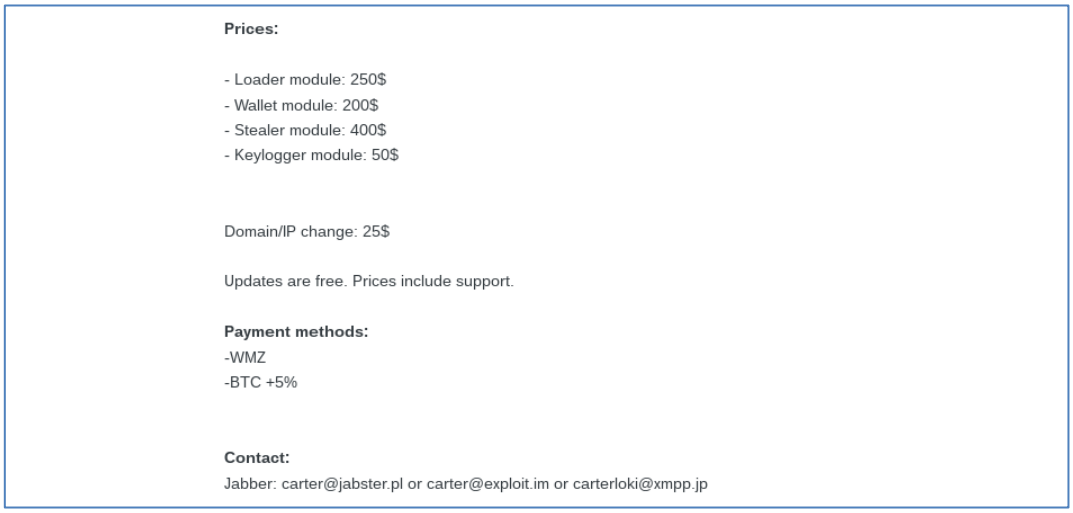
Loki Bot - Resident Loader and Password & Coin Wallet Stealer

Loki Bot is resident loader and password and cryptocoin-wallet stealer. It comes with wallet checker (coin inspector, read below). It can steal passwords from browsers, ftp/ssh, e-mail and poker clients. Written in C++. Works on Windows 2003, 2008, 2012, XP, Vista, 7, 8, 8.1, 10 and Linux. UAC bypass.

Loader features:
- Startup (resident loader)
- Download & Run (exe | dll)
- Download & Drop
- Update bot
- Uninstall bot
- Get password from bot per request
- Geotargetting, load to selected country
- Set reconnect interval
- Get screenshot

Figura 1. Lanzamiento de LokiBot en 2015

El paquete a la venta dentro del modelo de *Malware as a Service* (MaaS) se podía personalizar con los siguientes módulos, sujetos a las tasas establecidas por Carter.



Prices:

- Loader module: 250$
- Wallet module: 200$
- Stealer module: 400$
- Keylogger module: 50$

Domain/IP change: 25$

Updates are free. Prices include support.

Payment methods:
-WMZ
-BTC +5%

Contact:
Jabber: carter@jabster.pl or carter@exploit.im or carterloki@xmpp.jp

Figura 2. Módulos ofertados con LokiBot

Actualmente el usuario Carter se encuentra marcado como timador (*ripper* en inglés o кидала en ruso).

5. VERSIÓN CRACKEADA

Si bien aún sería posible observar muestras originales de LokiBot en campañas actuales, la gran mayoría de actores que están usando esta herramienta recurren a la versión *crackeada*. Desde que a finales de 2016 se liberaran las herramientas capaces de parchear binarios de LokiBot junto con el panel de control, sería correcto afirmar que LokiBot pasó de MaaS a *commodity* malware, ofreciendo a un variado rango de actores la posibilidad de incorporar este *stealer* a sus operaciones de forma gratuita.

Para generar binarios *crackeados* que apunten al panel de control deseado, la herramienta "Loki stealer get password" se encargaría automáticamente de hacer uso del *builder* con el panel de control que se indique en "Path Gate".

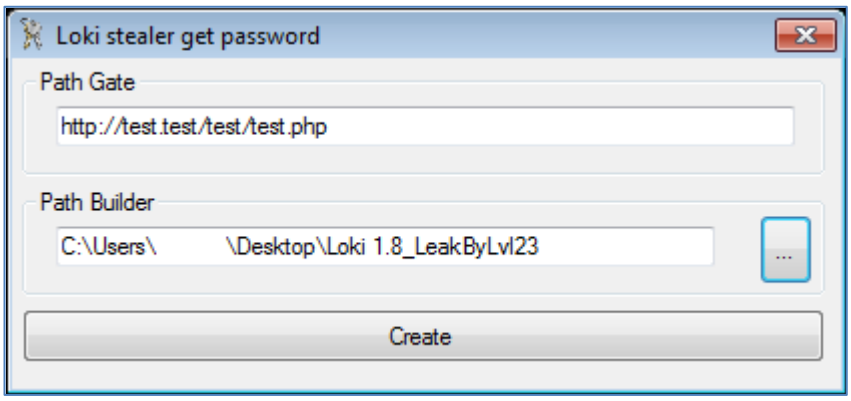


Figura 3. Herramienta "Loki stealer get pasword"

Si se conoce la contraseña del *builder*, se puede utilizar directamente esta herramienta especificando la URL del panel de control a continuación de la contraseña por línea de comandos.

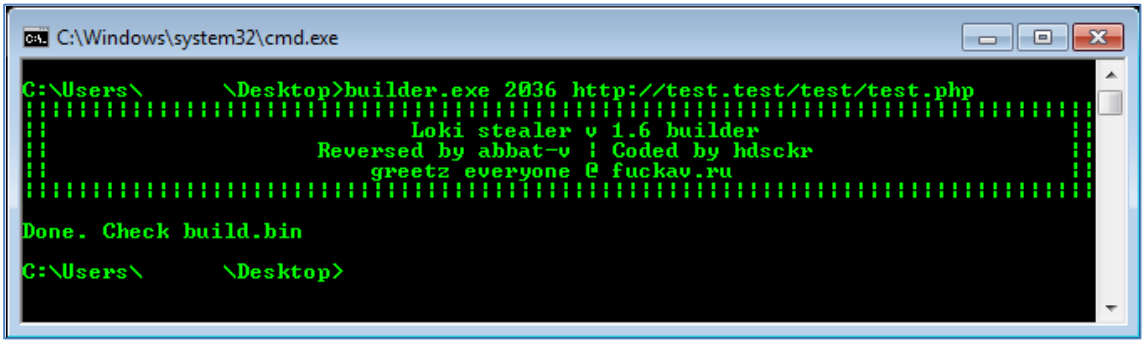


Figura 4. *Builder* generador de muestras crackeadas

Finalmente, el binario generado apuntaría al panel de control donde el actor desplegaría el *back-end* que se facilita en el kit.

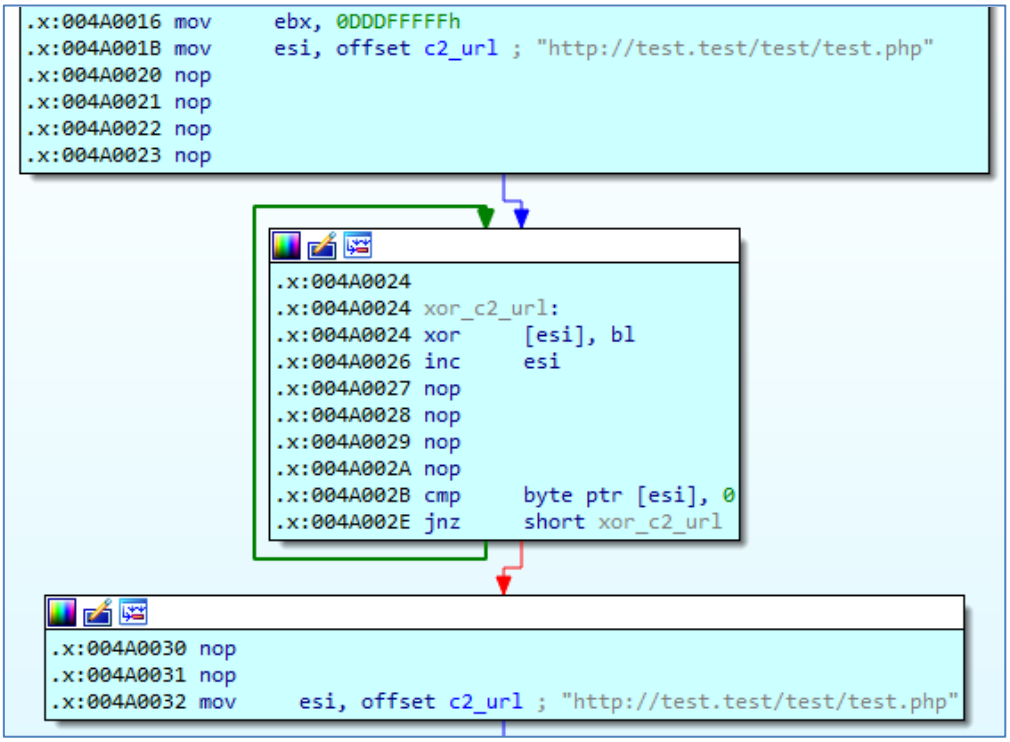


Figura 5. Test URL especificada en el *builder*

La versión *crackeada* muestra ciertas desventajas respecto de las muestras originales. Aunque se entrará en detalles a lo largo del informe, conviene adelantar que el parche aplicado por las herramientas mencionadas rompe la posibilidad de que la muestra adquiera persistencia y limita el número de direcciones del panel de control a sólo una.

6. CARACTERÍSTICAS DEL CÓDIGO DAÑINO

El código dañino es capaz de realizar las siguientes acciones.

- Exfiltrar información del usuario y sistema comprometido.
- Descargar payloads adicionales.
- Recibir actualizaciones.
- Desinstalarse.
- Enviar tareas a los *bots* en base a su localización.

7. PROCESO DE INFECCIÓN

El proceso de infección de la muestra analizada se puede dividir en cinco apartados, que se detallan a continuación.

7.1 PACKER .NET

Con el fin de ocultar el código dañino a soluciones de seguridad e investigadores, el binario distribuido en la campaña de LokiBot a analizar se encuentra protegido por un *packer* desarrollado bajo el *framework* .NET.

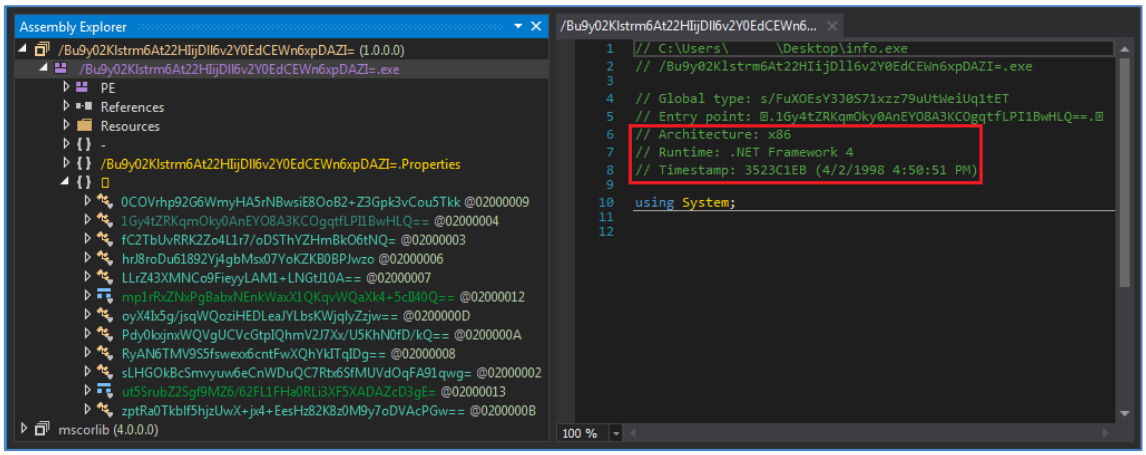


Figura 6. Binario inicial

La *decompilación* del código del *packer* muestra una fuerte ofuscación. Esta capa de seguridad añadida para prevenir la ingeniería inversa no es de uso exclusivo en software dañino, sino un método de proteger la propiedad intelectual incluida en programas legítimos y complicar los procesos de cracking.

Tras correr el código del *packer*, **el payload será inyectado en el binario legítimo RegAsm.exe**.

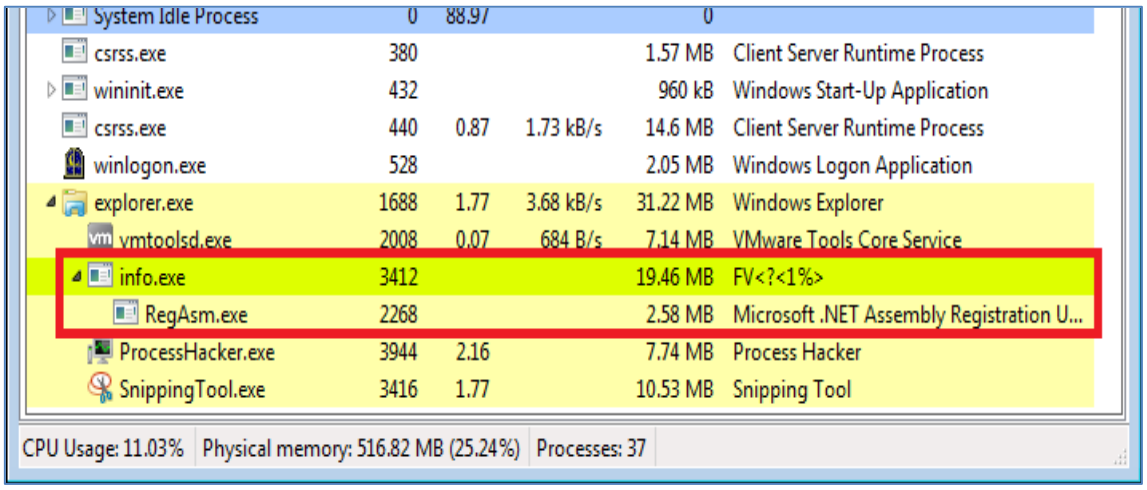


Figura 7. Inyección en RegAsm.exe

7.2 CONTROL DE INSTANCIAS

Antes de comenzar con las tareas de recolección de información del sistema infectado, el código dañino tratará de asegurar que una única instancia del malware se encuentre corriendo en el sistema. Para ello se creará un mutex cuyo nombre se derivará del valor del Machine GUID (*Globally Unique IDentifier*). Si el mutex ya existe, el nuevo proceso del código dañino concluye sin mayor afectación.

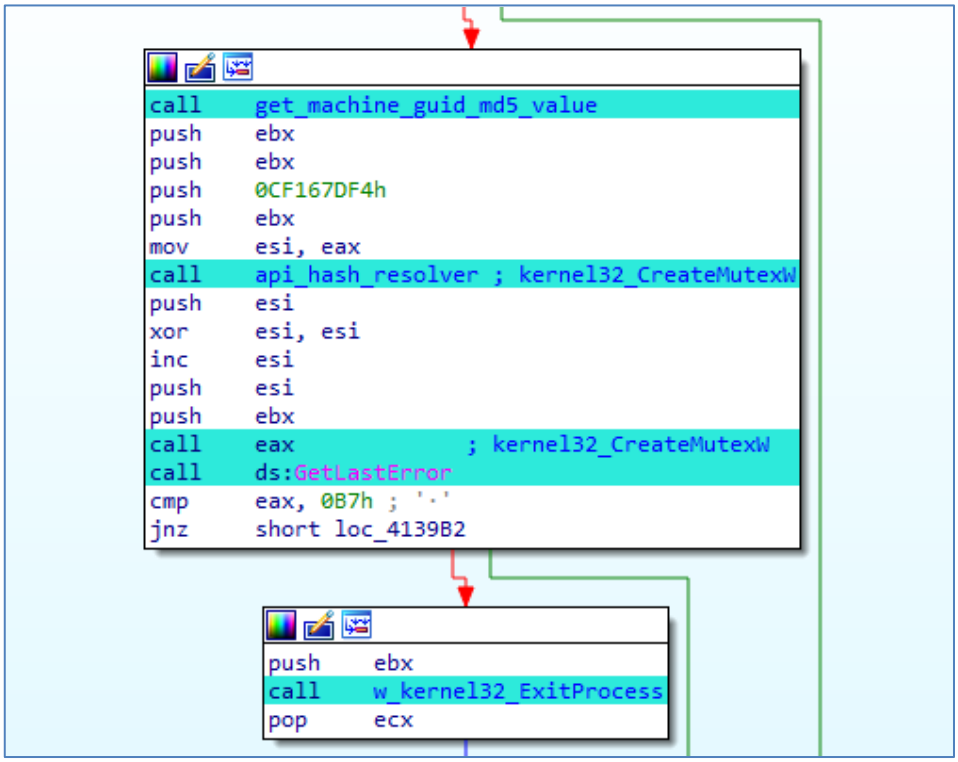


Figura 8. Control de instancias mediante mutex

El proceso de generación del nombre del mutex consiste en calcular el hash MD5 del valor del Machine GUID, para posteriormente usar los 24 primeros caracteres, en mayúsculas y UTF-16, para el nombre del mutex.

De tal manera, si el Machine GUID de un sistema mostrase el identificador:

f1ca7162-f4b7-41b2-11f8-e8dd710c412c

Su hash MD5 devolvería el valor **E3714F7C54A2C9421CC05F0364DD0766**, cuyos primeros 24 caracteres, **E3714F7C54A2C9421CC05F03**, darían nombre al mutex para ese sistema en concreto.

7.3 EXFILTRACIÓN DE INFORMACIÓN

Después de asegurar que sólo una instancia del código dañino se encuentre corriendo en el sistema, el código dañino tratará de recolectar información sensible de diversas aplicaciones para enviarla al panel de control.

La exfiltración se realizará en dos pasos. El primero de ellos tratará de recoger información de los tipos de programas que se listan a continuación, para enviarla en una primera petición al panel de control.

Programas de los que extraer información sensible	
Navegadores	Clientes de FTP, SSH y VNC
Clientes de correo electrónico	Clientes de mensajería instantánea
Gestores de contraseñas	Gestores de notas
Clientes de Póker	

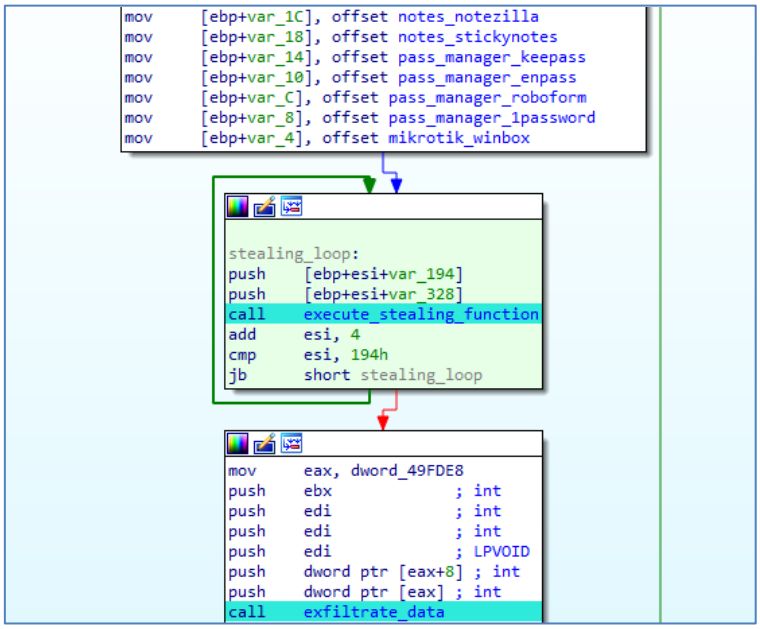


Figura 9. Extracción de información de aplicaciones

El listado completo de las aplicaciones de las que se tratará de recabar información se recoge en las siguientes tablas.

Navegadores			
360Browser	Chromium	Chromodo Browser	Citrio Browser
Comodo Dragon	Comodo IceDragon	CoolNovo	Coowon Browser
Cyberfox	Epic Privacy Browser	Google Chrome	Internet Explorer
Iridium	K-Meleon	Maxthon Browser	Mozilla Firefox
Mustan Browser	Opera	Orbitum	Pale Moon
Rambler Nichrome	RockMelt	Safari	SeaMonkey
Sleipnir Browser	Superbird Browser	Titan Browser	Torch Browser
Vivaldi	Waterfox	Yandex Browser	

Clientes de FTP, SSH y VNC			
32Bit FTP	AbleFTP	ALFTP	Automize
BitKinex	Bitvise	Blaze Ftp	ClassicFTP
CyberDuck	DeluxeFTP	Easy FTP	ExpanDrive
FAR Manager	FileZilla	FlashFXP	Fling
FreshFTP	FTP Navigator	FTP Now	FTP Rush
FTPBox	FTPGetter	FtpInfo	FTPShell
FullSync	GoFTP	JaSFtp	Kitty
Linas FTP	Maxthon browser	mSecure Wallet	MyFTP
NetDrive	NETFile	Nexus File	NovaFTP
NppFTP	OdinSecure FTP Expert	Putty	RealVNC
Remmina RDP	SherrodFTP	SmartFTP	Staff-FTP
Syncovery	TightVNC	Total Commander	UltraFXP
Vandyke SecureFX	WinFTP	WinSCP	WS_FTP
Xftp			

Clientes de correo electrónico			
Becky! Internet Mail	Checkmail	FossaMail	Foxmail
Gmail Notifier Pro	Incredimail	MailSpeaker	Mozilla Thunderbird
Opera Mail	Outlook	Pocomail	Postbox
Softwarenetz Mailer	Trojita	TrulyMail	yMail

Cliente de mensajería instantánea
Pidgin

Gestores de contraseñas	
1Password	EnPass
KeePass	RoboForm

Gestores de notas	
NoteFly	Notezilla
Stickies	StickyNotes
StickyPad	To-Do Desklist

Clientes de Póker	
Full Tilt Poker	PokerStars

En el segundo y último paso, se tratarán de extraer credenciales de Windows almacenadas en el sistema, que se enviarían en una segunda petición al panel de control.

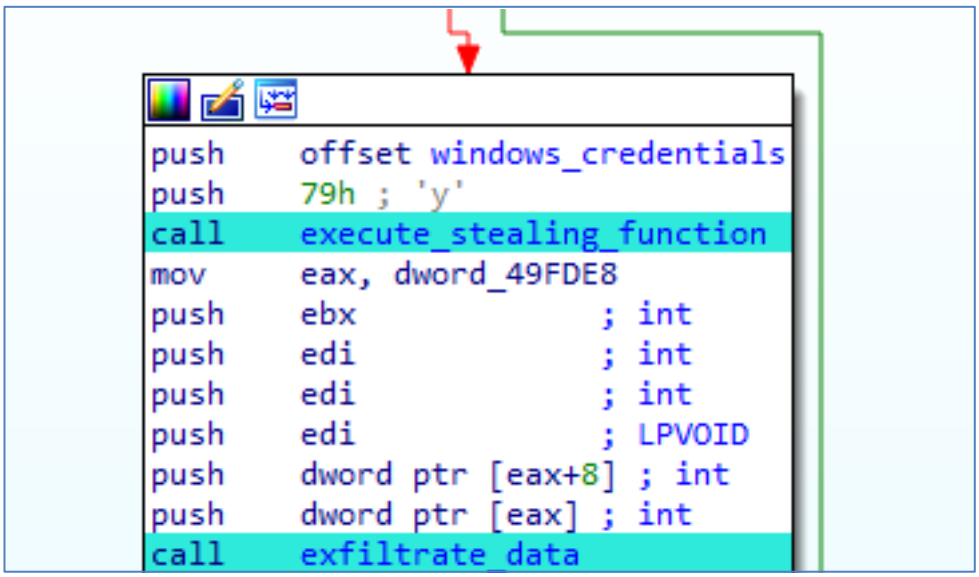


Figura 10. Extracción de credenciales de Windows

7.4 INSTALACIÓN

En el directorio **%AppData%** se localizará el directorio de instalación del código dañino, donde se copiará el ejecutable objeto de análisis junto con dos ficheros, que se crearán para asistir con el proceso de exfiltración de información.

En este punto también se tratará de lograr persistencia para asegurar la ejecución del malware entre reinicios del sistema. Sin embargo, debido al funcionamiento del *builder*, las muestras *crackeadas* pierden esta posibilidad de adquirir persistencia.

El método elegido sería crear una clave de registro en la ruta que se muestra a continuación, apuntando al ejecutable en su directorio de instalación en **%AppData%**.

Software\Microsoft\Windows\CurrentVersion\Run

El problema se encuentra en que la función encargada de descifrar la ruta del registro de Windows se encuentra parcheada por el *builder* para devolver siempre la URL del panel de control. En la siguiente captura se puede observar cómo quedan restos de la clave de registro tras escribir encima la URL que se especificó en el *builder*.

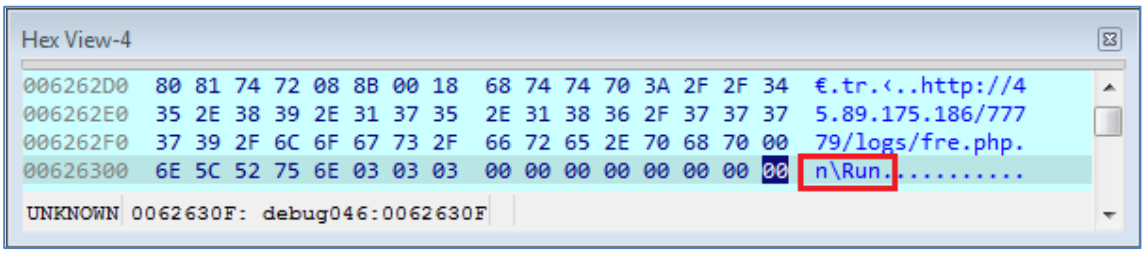


Figura 11. Clave de registro sobrescrita por la URL del panel de control

El nombre del directorio, así como el de los ficheros en su interior, están relacionados con el valor del MD5 que se calculó a partir del Machine GUID para generar el mutex.

Si el mutex tomaba los 24 primeros caracteres del MD5, el directorio será nombrado con los caracteres entre las posiciones 8 a 13 y los ficheros en su interior, con los caracteres entre las posiciones 13 a 18 todos ellos, aunque con distintas extensiones.

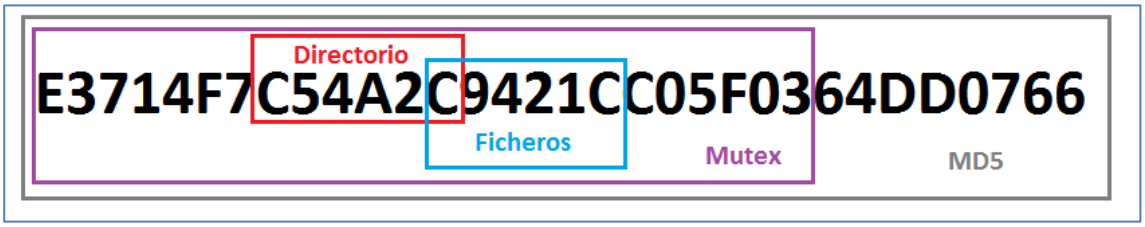


Figura 12. Generación del nombre de los artefactos de LokiBot

La tabla que se muestra a continuación recoge el total de ficheros que se pueden encontrar en el directorio de instalación del código dañino.

Fichero	Descripción
.exe	Copia del código dañino
.hdb	Fichero creado para controlar qué información ha sido ya enviada al panel de control (*hashes database*)
.lck	Fichero creado para prevenir conflictos de acceso a determinados recursos (*lock*)

Finalmente recalcar que **el directorio se oculta al usuario** otorgándole los atributos necesarios para producir tal efecto y que los nombres, tanto del directorio como de los ficheros, son únicos para cada máquina.

7.5 SOLICITUD DE COMANDOS A EJECUTAR

Tras el intento de adquisición de persistencia, el código dañino contacta por tercera vez con el panel de control. En esta ocasión la comunicación se realiza con el objetivo de solicitar al back-end comandos para ser ejecutados por el *bot*.

Desde el panel de control se puede solicitar al *bot* la ejecución de los siguientes comandos, que se procesarán en un nuevo hilo.

Comando	Descripción
0	Descargar y ejecutar EXE
1	Descargar y ejecutar DLL
2	Descargar y ejecutar DLL

Comando	Descripción
8	Borrar fichero HDB
10	Ejecutar procedimiento de exfiltración de información
14	Finalizar ejecución
15	Actualizar LokiBot
16	Cambiar frecuencia de peticiones
17	Eliminar LokiBot y finalizar ejecución

Aunque existe un comando específico para cambiar la frecuencia de estas peticiones, que por defecto serían realizadas en intervalos de 10 minutos, la funcionalidad no se encuentra en uso. Además, se ha observado que la frecuencia con la que se solicitarán comandos sería de 60 segundos.

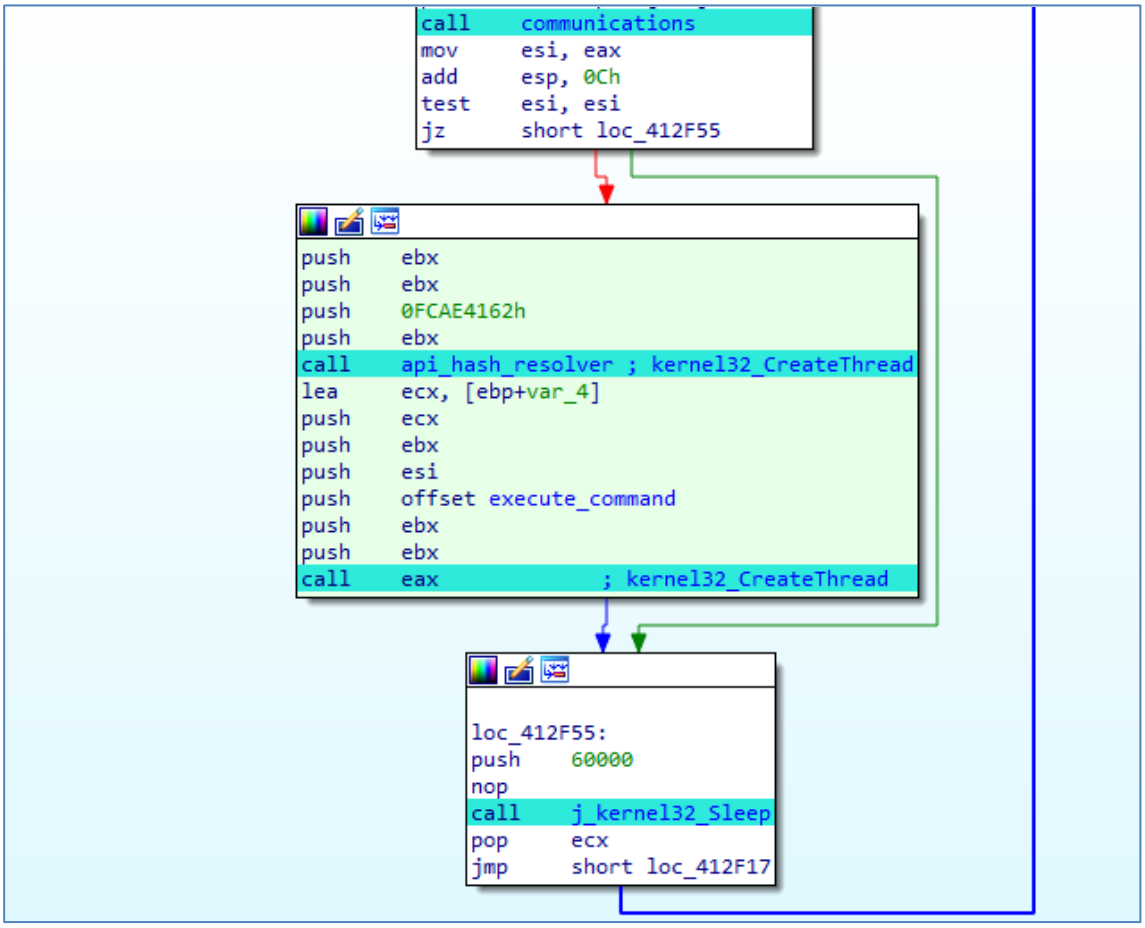


Figura 13. Los comandos recibidos se procesan en un nuevo hilo de ejecución

8. COMUNICACIONES

Tras haber introducido los puntos en los que las comunicaciones tendrán lugar, en este apartado se procede a resaltar ciertos detalles del proceso.

La muestra analizada sólo podrá intentar establecer comunicación con el panel de control que se muestra a continuación.

http://45.89.175.186/77779/logs/fre.php

Los servidores de mando y control que la muestra original incluía, cifrados bajo el algoritmo *Triple Data Encryption Standard* (3DES), no se podrán usar debido a que el parche destinado a *crackear* los ejecutables de LokiBot forzará a que siempre se devuelva la URL fijada por el *builder* cuando se llama a la función de descifrado 3DES. Por este mismo motivo, la clave de registro que se usaría para la persistencia es sobrescrita por el panel de control, puesto que también sería descifrada mediante 3DES, rompiendo de esta manera la persistencia.

El user-agent que utiliza el código dañino para la sesión de comunicación se lista a continuación.

Mozilla/4.08 (Charon; Inferno)

En el momento de realizar el análisis, el panel de control no se encontraba activo por lo que no se ha podido investigar en mayor detalle las posibles tareas disponibles para ejecutar por los *bots*.

9. CAMPAÑA OBJETO DE ANÁLISIS

Puesto que las fechas de compilación no ayudan a establecer un periodo orientativo en el que situar la campaña en la que fue utilizado el binario objeto de análisis, recurriendo a búsquedas en *Open Source Intelligence* (OSINT) y la fecha en la que el binario fue visto por primera vez, se puede hacer una estimación.

Mientras que el recurso **fre.php** es muy común en campañas de LokiBot, dado que los operadores del panel no suelen cambiarlo, el directorio **77779** se puede utilizar como punto de *pivoting* para encontrar campañas similares. La primera que se ha encontrado data del 16 de septiembre de 2019, publicada por “My online security” en Twitter y facilitando los ficheros involucrados en la campaña en el mismo tweet.

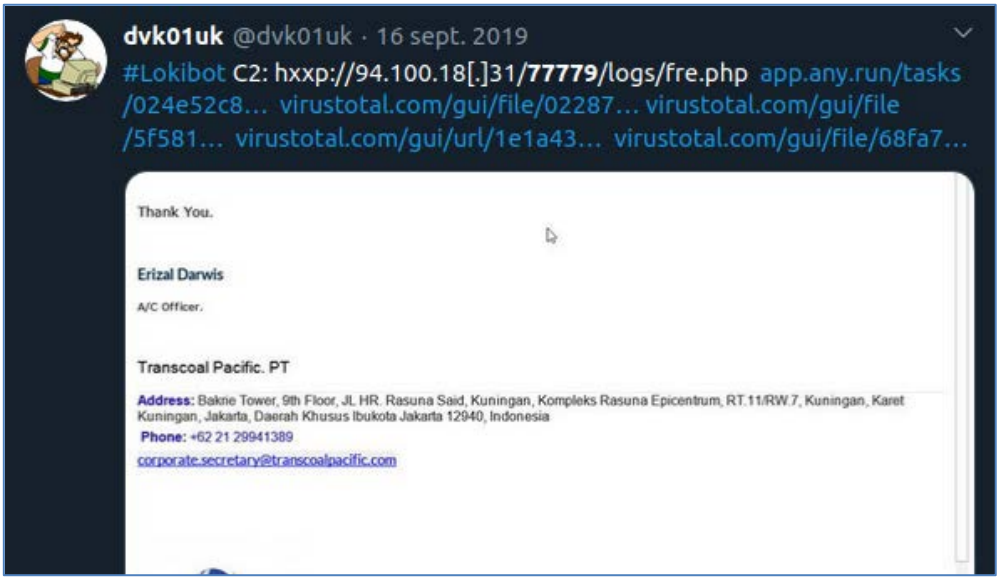


Figura 14. https://twitter.com/dvk01uk/status/1173518955161509889?s=20

Se ha encontrado otra campaña, publicada en Twitter también por el mismo usuario, que tuvo lugar el 25 de septiembre de 2019.

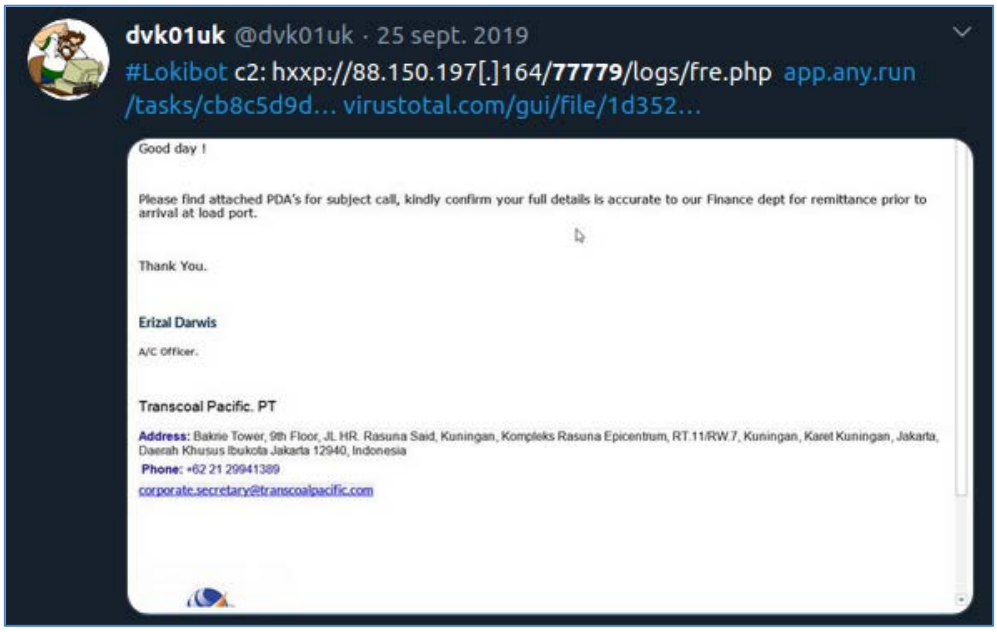


Figura 15. https://twitter.com/dvk01uk/status/1176800094764589056?s=20

Cabe destacar en este punto que tanto el binario analizado como los dos ejecutables listados en sendos tweets utilizan el mismo *packer* desarrollado en .NET y el mismo ofuscador de código.

Dado que la fuente se considera usualmente confiable y que la fecha en la que se vio por primera vez el binario objeto fue el 3 de septiembre de 2019, se consideraría probablemente cierto afirmar que, la campaña a la que pertenece el ejecutable analizado tuvo lugar a principios de septiembre o finales de agosto de 2019.

10. DESINFECCIÓN

Debido al bug del código dañino al tratar de establecer persistencia, un simple reinicio garantizaría el cese de su actividad en el equipo afectado.

Dada la naturaleza de la muestra analizada y debido a su capacidad de recibir ficheros adicionales, no se puede garantizar una limpieza completa del equipo tras interrumpir la ejecución del binario objeto de análisis.

11. REGLAS DE DETECCIÓN

11.1 REGLA YARA LOKIBOT

```
rule lokibot {
   meta:
      date = "2020-02-18"
      autor = "CCN"
   strings:
      $MachineGuid = "MachineGuid" ascii
      $msft_crypt = "SOFTWARE\\Microsoft\\Cryptography" ascii
      $aplib = "aPLib v1.01  -  the smaller the better :)" ascii
      $fuckav_ru = "Fuckav.ru" ascii
      $app_0 = "360Browser" wide
      $app_1 = "Comodo\\Dragon" wide
      $app_2 = "Google\\Chrome" wide
      $app_3 = "Nichrome" wide
      $app_4 = "RockMelt" wide
      $app_5 = "Chromium" wide
      $app_6 = "Titan Browser" wide
      $app_7 = "Torch" wide
      $app_8 = "Epic Privacy Browser" wide
      $app_9 = "Vivaldi" wide
      $app_10 = "Superbird" wide
   condition: uint16(0) == 0x5A4D and (all of them)
}
```

12. INDICADORES DE COMPROMISO

Binario inicial
ff3e61a10ae528487ed74feb73108a2e473d4582c5908f66643fdf002dc67e4c
Panel de control
http://45.89.175.186/77779/logs/fre.php
User-agent
Mozilla/4.08 (Charon; Inferno)
Directorio de instalación y ficheros
%AppData%\[A-F0-9]{6}
%AppData%\[A-F0-9]{6}\[A-F0-9]{6}.exe
%AppData%\[A-F0-9]{6}\[A-F0-9]{6}.hdb
%AppData%\[A-F0-9]{6}\[A-F0-9]{6}.lck