No.23 / marzo-abril 2015 ISSN: 1251478, 1251477

.Seguridad

Cultura de prevención para TI

23

Herramientas de detección

Investigación, desarrollo y acción

Dionaea | La alternativa Linux | IDS Raspbian | Buffer overflow | TICs y ciberterrorismo | Riesgos en Android

Contenido

.Seguridad
Cultura de prevención para TI

Editorial

Herramientas de detección

Investigación, desarrollo y acción

¿Sólo los que buscan encuentran? Al parecer en la seguridad informática esta frase tan popular no es tan simple ni del todo cierta.

Las amenazas inundan nuestros sistemas sin que podamos notarlo siquiera, la cuestión no está en buscarlas o no, sino en identificarlas, en detectar qué y cómo pone en riesgo nuestra información y nuestros procesos.

En esta edición te invitamos a conocer dos herramientas de detección, una de intrusos en redes inalámbricas y la otra, una prueba de concepto para captura de software malicioso.

Para los usuarios finales tenemos una excelente recomendación sobre cómo implementar Linux como su sistema operativo, también explicamos la vulnerabilidad buffer overflow, una de las más comunes a lo largo de los años, para los usuarios de Android tenemos una revisión sobre los riesgos en este sistema operativo y finalmente, te mostramos el panorama hacia el mundo del ciberterrorismo y su desarrollo a raíz del uso de las tecnologías de la información.

Somos conscientes de que no nos es posible combatir lo que no se conoce, para esto es necesario partir del conocimiento, de la investigación, después del desarrollo y finalmente de la acción. Aprenderemos nuevas lecciones y otras no, en fin, el ciclo comienza y vuelve a comenzar.

L.C.S Jazmín López Sánchez
Editora
Coordinación de Seguridad de la Información

.Seguridad
Cultura de prevención para TI

.Seguridad Cultura de prevención TI M.R. / Número 23 / marzo-abril 2015 / ISSN No. 1251478, 1251477 / Revista Bimestral, Registro de Marca 129829

PoC: Captura de *malware* con el *honeypot* Dionaea - Parte I

Jonathan Banfi Vázquez

En los últimos años se ha podido observar la evolución acelerada de las tecnologías de la información y comunicación a tal grado que al día de hoy conocemos y disfrutamos de los servicios que nos permiten acceder a la información que se genera en el mundo de manera rápida y sencilla. Sin embargo, este mismo desarrollo ha generado diversas amenazas, que si bien no son nuevas, se han adaptado perfectamente al tiempo y a las necesidades actuales, de tal manera que representan serios problemas para usuarios finales, para pequeñas y medianas empresas, grandes organizaciones, universidades y gobiernos.

Ante este panorama, los profesionales de la seguridad de la información se han visto en la necesidad de desarrollar habilidades y conocimiento que les permita estudiar y analizar el comportamiento, la estructura, los riesgos y el impacto que estas amenazas pueden provocar.

El término que engloba al conjunto de elementos al que nos referimos es ***malware***, acrónimo de *malicious software*, dentro del cual se incluyen virus, gusanos, troyanos, puertas traseras, *bots, rootkits, spyware, rogues, ransomware, downloaders, droppers*, entre otros.

El objetivo de este artículo es mostrar precisamente la captura de este tipo de amenazas con el *honeypot* Dionaea en un entorno virtual controlado y sin salida a Internet para evitar propagar la amenaza. En estas publicaciones describiré todo lo necesario para instalar y configurar Dionaea de forma que nos permita realizar satisfactoriamente la prueba de concepto.

En esta primera entrega nos enfocaremos en conocer e implementar las herramientas necesarias que se utilizarán en el laboratorio. Posteriormente, en una segunda edición,

mostraremos la configuración de las herramientas y la captura de gusanos informáticos.

Requerimientos de software

Máquinas virtuales	*Honeypots*	Herramientas de monitoreo	*Malware*
Debian 7	Dionaea	Process Explorer	Gusano445
Windows XP	Honeyd	TCPView	

Dionaea

Es un *honeypot* de baja interacción (emula servicios), sucesor del proyecto **Nepenthes**, escrito en C pero que además incorpora Python como lenguaje de *scripting*, utiliza la biblioteca **libemu** para emular la ejecución de instrucciones Intel x86 y detectar *shellcodes*. Además, cuenta con soporte para IPv6 y protocolo TLS.

Honeyd

Es una herramienta de código abierto para implementar *honeypots* de baja interacción, fue desarrollada por Niels Provos, miembro del Proyecto Honeynet. Simula la presencia de *hosts* en una red, es decir, proporciona a un solo equipo todo el espectro de direcciones IPv4). Intercepta únicamente conexiones dirigidas a direcciones IP no existentes en nuestra red, es decir, excluye las direcciones IP del segmento donde se ha implementado.

Puede configurarse para correr ciertos servicios vulnerables y redireccionar las conexiones de red a un *listener* o servidor local.

Process Explorer

Es una herramienta de Microsoft que forma parte de la suite de SysInternals y muestra información de los procesos que se ejecutan en el sistema, así como las DLL (*Dynamic Link Library*) asociadas a cada uno de ellos.

TCPView

Es una herramienta de Microsoft que forma parte de la suite de SysInternals y ofrece una lista del estado de las conexiones y puertos TCP/UDP del sistema, así como el identificador de su proceso asociado (PID).

Gusano

Aplicación informática con la capacidad de autoreplicarse e invadir los equipos de cómputo de una red, busca explotar alguna vulnerabilidad en los servicios para trasladarse de un *host* a otro con el fin de realizar alguna acción maliciosa.

Implementación

La configuración de los repositorios en la máquina virtual Debian 7 es muy importante, pero no es la finalidad de este artículo explicar este proceso, sin embargo se recomienda este enlace al lector, donde puede generar una lista personalizada.

Una vez actualizada la lista de repositorios, lo siguiente es instalar los *honeypots* que necesitaremos.

Honeyd

1) Crear directorios de trabajo:

```
root@MalwareAnalysisLab:/home/malware# mkdir /opt/honeyd
root@MalwareAnalysisLab:/home/malware# mkdir /opt/honeydPackages
root@MalwareAnalysisLab:/home/malware# cd /opt/honeydPackages
root@MalwareAnalysisLab:/opt/honeydPackages#
```

2) Instalar dependencias:

```
root@MalwareAnalysisLab:/opt/honeydPackages# apt-get -y install git autoconf aut
ogen bison flex nmap libdnet libdnet-dev libedit-dev libdumbnet-dev libtool zlib
1g-dev libpcre3 libpcre3-dev gettext
```

3) Instalar Libevent:

```
root@MalwareAnalysisLab:/opt/honeydPackages# wget https://github.com/downloads/l
ibevent/libevent/libevent-2.0.21-stable.tar.gz \
> && tar -xzvf libevent-2.0.21-stable.tar.gz \
> && pushd libevent-2.0.21-stable \
> && ./configure --prefix=/opt/honeyd \
> && make \
> && make install \
> && popd
```

4) Instalar Libpcap:

```
root@MalwareAnalysisLab:/opt/honeydPackages# wget http://www.tcpdump.org/release
/libpcap-1.1.1.tar.gz \
> && tar xvfz libpcap-1.1.1.tar.gz \
> && pushd libpcap-1.1.1 \
> && ./configure \
> && make \
> && make install \
> && popd
```

5) Instalar Honeyd:

```
root@MalwareAnalysisLab:/opt/honeydPackages# git clone git://github.com/DataSoft
/Honeyd.git honeyd \
> && pushd honeyd \
> && autoreconf -vi \
> && ./configure --prefix=/opt/honeyd --with-libevent=/opt/honeyd \
> && make \
> && make install \
> && popd
```

6) Actualizar bibliotecas:

```
root@MalwareAnalysisLab:/opt/honeydPackages# ldconfig
root@MalwareAnalysisLab:/opt/honeydPackages#
```

7) Verificar la creación del binario "honeyd":

```
root@MalwareAnalysisLab:/opt/honeydPackages# cd /opt/honeyd/bin/
root@MalwareAnalysisLab:/opt/honeyd/bin#
root@MalwareAnalysisLab:/opt/honeyd/bin# ls
event_rpcgen.py  honeyd  honeydctl  honeydstats  hsniff
root@MalwareAnalysisLab:/opt/honeyd/bin#
root@MalwareAnalysisLab:/opt/honeyd/bin# file honeyd
honeyd: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically li
nked (uses shared libs), for GNU/Linux 2.6.26, BuildID[sha1]=0xda132539d84243767
a2ed147fb10d3d637bb606e, not stripped
root@MalwareAnalysisLab:/opt/honeyd/bin#
```

Dionaea

1) Crear directorios de trabajo:

```
root@MalwareAnalysisLab:/home/malware# mkdir /opt/dionaea
root@MalwareAnalysisLab:/home/malware# mkdir /opt/dionaeaPackages
root@MalwareAnalysisLab:/home/malware# cd /opt/dionaeaPackages
root@MalwareAnalysisLab:/opt/dionaeaPackages#
```

2) Instalar dependencias:

```
root@MalwareAnalysisLab:/opt/dionaeaPackages# apt-get -y install git libglib2.0-
dev libssl-dev libcurl4-openssl-dev libreadline-dev libsqlite3-dev python-dev li
btool automake autoconf build-essential subversion git-core flex bison pkg-confi
g libgc-dev libglib2.0 libglib2.0-0 libc6-dev gettext libffi-dev readline-common
 python2.6-dev libcurl3 libcurl3-dev sqlite3 libsqlite3-dev libreadline6 libread
line6-dev php5-common php5-cgi php5-cli php5-fpm php5-sqlite libxml2-dev libxslt
1-dev libc6-dev-amd64 g++-multilib lib64stdc++6 lib64z1-dev
```

3) Instalar Libudns:

```
root@MalwareAnalysisLab:/opt/dionaeaPackages# wget http://www.corpit.ru/mjt/udns
/libudns0_0.1-1_i386.deb \
> && wget http://www.corpit.ru/mjt/udns/libudns-dev_0.1-1_i386.deb \
> && dpkg -i libudns0_0.1-1_i386.deb \
> && dpkg -i libudns-dev_0.1-1_i386.deb
```

4) Instalar Glib:

```
root@MalwareAnalysisLab:/opt/dionaeaPackages# wget http://ftp.gnome.org/pub/gnom
e/sources/glib/2.20/glib-2.20.4.tar.bz2 \
> && tar xvfj glib-2.20.4.tar.bz2 \
> && pushd glib-2.20.4 \
> && ./configure --prefix=/opt/dionaea/ \
> && make \
> && make install \
> && popd
```

5) Instalar Liblcfg:

```
root@MalwareAnalysisLab:/opt/dionaeaPackages# git clone git://git.carnivore.it/l
iblcfg.git liblcfg \
> && pushd liblcfg/code \
> && autoreconf -vi \
> && ./configure --prefix=/opt/dionaea \
> && make \
> && make install \
> && popd
```

6) Instalar Libemu:

```
root@MalwareAnalysisLab:/opt/dionaeaPackages# git clone git://git.carnivore.it/l
ibemu.git libemu \
> && pushd libemu \
> && autoreconf -vi \
> && ./configure --prefix=/opt/dionaea \
> && make \
> && make install \
> && popd
```

7) Instalar Libgc:

```
root@MalwareAnalysisLab:/opt/dionaeaPackages# wget https://launchpad.net/ubuntu/
+archive/primary/+files/libgc_6.8.orig.tar.gz \
> && tar xvfz libgc_6.8.orig.tar.gz \
> && pushd gc6.8 \
> && ./configure --prefix=/opt/dionaea \
> && make \
> && make install \
> && popd
```

8) Instalar Libev:

```
root@MalwareAnalysisLab:/opt/dionaeaPackages# wget --user-agent="Mozilla/5.0 (Wi
ndows; U; Windows NT 5.1; en-US; rv:1.8.1.6) Gecko/20070725 Firefox/2.0.0.6" htt
p://dist.schmorp.de/libev/Attic/libev-4.04.tar.gz \
> && tar xvfz libev-4.04.tar.gz \
> && pushd libev-4.04 \
> && ./configure --prefix=/opt/dionaea \
> && make \
> && make install \
> && popd
```

9) Instalar Python:

```
root@MalwareAnalysisLab:/opt/dionaeaPackages# wget --no-check-certificate https:
//www.python.org/ftp/python/3.2.2/Python-3.2.2.tgz \
> && tar xvfz Python-3.2.2.tgz \
> && pushd Python-3.2.2 \
> && ./configure --enable-shared --prefix=/opt/dionaea --with-computed-gotos --e
nable-ipv6 LDFLAGS="-Wl,-rpath=/opt/dionaea/lib/ -L/usr/lib/x86_64-linux-gnu/" \
> && make \
> && make install \
> && popd
```

10) Instalar Cython:

```
root@MalwareAnalysisLab:/opt/dionaeaPackages# git clone https://github.com/cytho
n/cython.git cython \
> && pushd cython \
> && /opt/dionaea/bin/python3 setup.py install \
> && popd
```

11) Instalar Lxml:

```
root@MalwareAnalysisLab:/opt/dionaeaPackages# git clone https://github.com/lxml/
lxml.git lxml \
> && pushd lxml \
> && /opt/dionaea/bin/2to3 -w src/lxml/html/_diffcommand.py \
> && /opt/dionaea/bin/2to3 -w src/lxml/html/_html5builder.py \
> && /opt/dionaea/bin/python3.2 setup.py build \
> && /opt/dionaea/bin/python3.2 setup.py install \
> && popd
```

12) Instalar Udns:

```
root@MalwareAnalysisLab:/opt/dionaeaPackages# wget http://www.corpit.ru/mjt/udns
/old/udns_0.0.9.tar.gz \
> && tar xvfz udns_0.0.9.tar.gz \
> && pushd udns-0.0.9 \
> && ./configure \
> && make shared \
> && cp udns.h /opt/dionaea/include/ \
> && cp *.so* /opt/dionaea/lib/ \
> && pushd /opt/dionaea/lib \
> && ln -s libudns.so.0 libudns.so \
> && popd \
> && popd
```

13) Instalar C-ares:

```
root@MalwareAnalysisLab:/opt/dionaeaPackages# wget http://c-ares.haxx.se/downloa
d/c-ares-1.9.1.tar.gz \
> && tar xvfz c-ares-1.9.1.tar.gz \
> && pushd c-ares-1.9.1 \
> && ./configure --prefix=/opt/dionaea \
> && make \
> && make install \
> && popd
```

14) Instalar Curl:

```
root@MalwareAnalysisLab:/opt/dionaeaPackages# wget http://curl.haxx.se/download/
curl-7.28.1.tar.bz2 \
> && tar xvfj curl-7.28.1.tar.bz2 \
> && pushd curl-7.28.1 \
> && ./configure --prefix=/opt/dionaea --enable-ares=/opt/dionaea \
> && make \
> && make install \
> && popd
```

15) Instalar Libpcap:

```
root@MalwareAnalysisLab:/opt/dionaeaPackages# wget http://www.tcpdump.org/releas
e/libpcap-1.1.1.tar.gz \
> && tar xvfz libpcap-1.1.1.tar.gz \
> && pushd libpcap-1.1.1 \
> && ./configure \
> && make \
> && make install \
> && popd
```

16) Instalar Openssl:

```
root@MalwareAnalysisLab:/opt/dionaeaPackages# git clone git://git.openssl.org/op
enssl.git openssl \
> && pushd openssl \
> && ./Configure shared --prefix=/opt/dionaea linux-x86_64 \
> && make SHARED_LDFLAGS=-Wl,-rpath,/opt/dionaea/lib \
> && make \
> && make install \
> && popd
```

17) Instalar Dionaea:

```
root@MalwareAnalysisLab:/opt/dionaeaPackages# git clone git://git.carnivore.it/d
ionaea.git dionaea \
> && pushd dionaea \
> && autoreconf -vi \
> && ./configure --with-lcfg-include=/opt/dionaea/include/ --with-lcfg-lib=/opt/
dionaea/lib/ --with-python=/opt/dionaea/bin/python3.2 --with-cython-dir=/opt/dio
naea/bin --with-udns-include=/opt/dionaea/include/ --with-udns-lib=/opt/dionaea/
lib/ --with-emu-include=/opt/dionaea/include/ --with-emu-lib=/opt/dionaea/lib/ -
-with-gc-include=/usr/include/gc --with-ev-include=/opt/dionaea/include --with-e
v-lib=/opt/dionaea/lib --with-nl-include=/opt/dionaea/include --with-nl-lib=/opt
/dionaea/lib/ --with-curl-config=/usr/bin/ --with-pcap-include=/opt/dionaea/incl
ude --with-pcap-lib=/opt/dionaea/lib/ --with-glib=/opt/dionaea --prefix=/opt/dio
naea \
> && make \
> && make install \
> && popd
```

18) Verificar la creación del binario "dionaea":

```
root@MalwareAnalysisLab:/opt/dionaeaPackages# cd /opt/dionaea/bin/
root@MalwareAnalysisLab:/opt/dionaea/bin#
root@MalwareAnalysisLab:/opt/dionaea/bin# ls
2to3          dionaea          idle3.2           python3.2m
2to3-3.2      glib-genmarshal  lcfg              python3.2m-config
c_rehash      glib-gettextize  openssl           python3-config
curl          glib-mkenums     pydoc3            readlogsqltree
curl-config   gobject-query    pydoc3.2          scprofiler
cygdb         gtester          python3           sctest
cython        gtester-report   python3.2
cythonize     idle3            python3.2-config
root@MalwareAnalysisLab:/opt/dionaea/bin#
root@MalwareAnalysisLab:/opt/dionaea/bin# file dionaea
dionaea: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically l
inked (uses shared libs), for GNU/Linux 2.6.26, BuildID[sha1]=0xf43ef7253602767b
b437c5620e77c23113039bc1, not stripped
root@MalwareAnalysisLab:/opt/dionaea/bin#
```

Hasta este momento se mostró una forma de instalar en un sistema operativo Debian 7 las herramientas Honeyd y Dionaea que se usarán en conjunto para capturar *malware* en un entorno virtual controlado y sin salida a Internet. Cabe mencionar que si su empresa se dedica a la seguridad y desean contribuir con firmas antivirus o *sandbox* públicas con nuevas amenazas, pueden implementar Dionaea (como lo describe este artículo) en sus equipos destinados para análisis de tráfico de red y capturar gusanos informáticos. La instalación de Honeyd sólo será para realizar la PoC, no se necesita en sensores de tráfico de red.

A continuación se muestran algunos comandos y directorios importantes para el uso de Dionaea:

- Asociar direcciones IP públicas a una interfaz de red en Linux (entre más direcciones disponibles tengan para asignar a Dionaea es mejor, ya que aumenta la probabilidad de que algún gusano informático de muchos que hay en Internet solicite esa dirección):

```
ip addr add <dirección_IP_pública>/32 dev
<interfaz_de_red>
```

- Iniciar el servicio de Dionaea:

```
/opt/dionaea/bin/dionaea -D -r /opt/
dionaea -w /opt/dionaea -p /opt/dionaea/
var/dionaea.pid -l all,-debug -L '*'
```

Directorio que almacena las muestras capturadas:

```
/opt/dionaea/var/dionaea/binaries/
```

Por falta de espacio, dejaremos el análisis de la PoC para la siguiente edición de la revista.

Si quieres saber más consulta:

- Códigos maliciosos
- Conficker en México
- Códigos maliciosos en dispositivos móviles
- Honeypots: alternativas para analizar códigos maliciosos

Jonathan Banfi Vázquez

Ingeniero en Computación (UNAM), fue miembro de la sexta generación del Programa de Becas de Formación Seguridad Informática de UNAM-CERT. Labora desde 2012 en la CSI/UNAM-CERT en el área de Análisis de software malicioso.

Ha impartido clases como profesor de asignatura en la Facultad de Ciencias de la UNAM, además de varios cursos y capacitaciones relacionados con seguridad informática.

La alternativa Linux

Sergio Andrés Becerril

Todos conocen a Windows

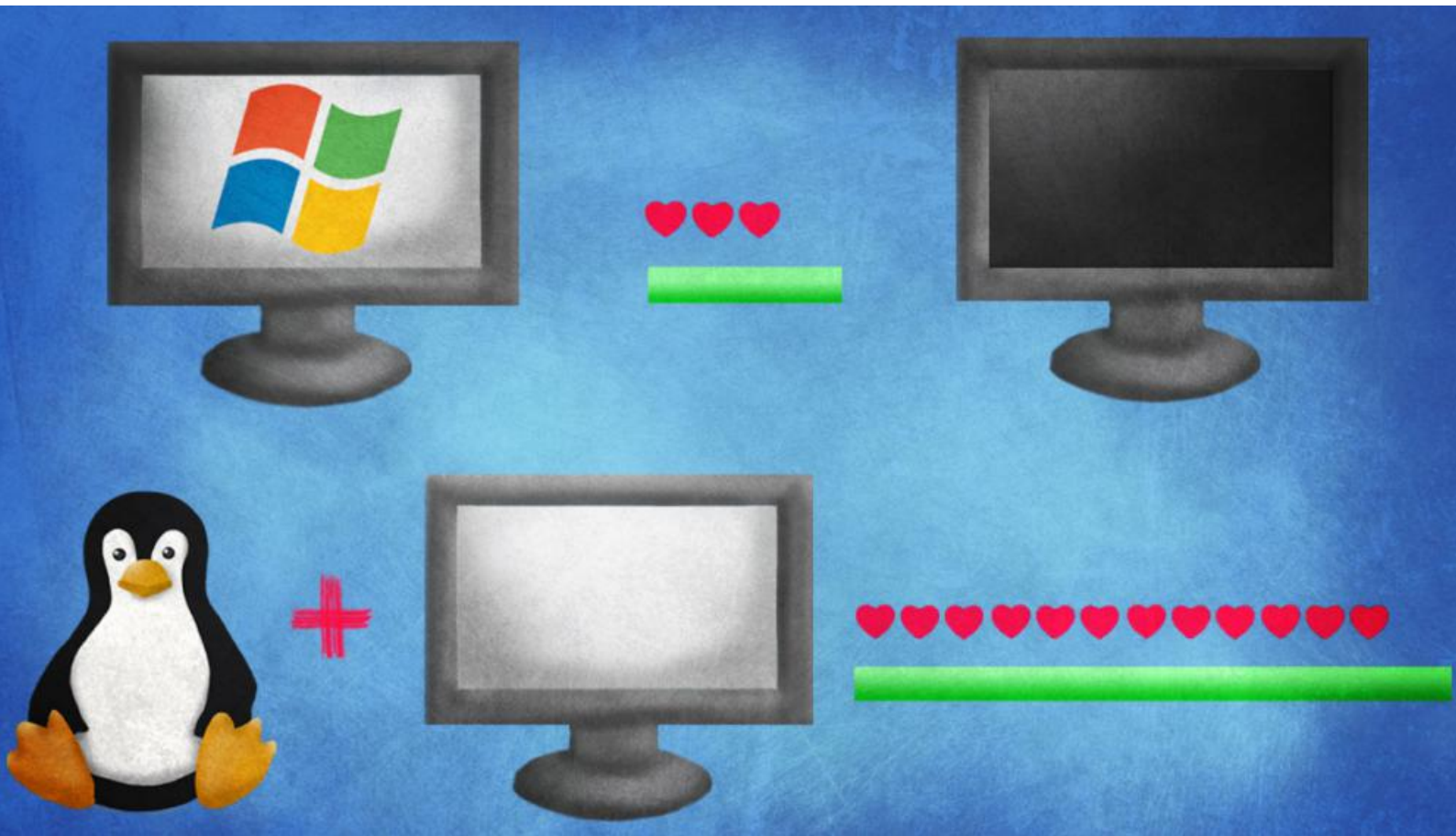
Windows es un **sistema operativo**; es decir, un programa que te permite ejecutar a **otros** programas, administrar tus archivos, conectarte a la red inalámbrica, etc. Con Windows puedes hacer uso de tu computadora y cualquier dispositivo que conectes a ella.

Si eres como la mayoría de nosotros, prácticamente todas las computadoras con las que trabajas diariamente utilizan Windows. Casi todas las computadoras nuevas tienen Windows instalado de fábrica, listo para usarlo en cuanto la saques de su empaque. ¡Incluso los cajeros automáticos usan Windows!

Probablemente sabes también que Windows no es el *único* sistema operativo que existe. Por ejemplo, en dispositivos móviles (que también necesitan un sistema operativo) existe **iOS**, el sistema operativo para móviles de Apple que puedes encontrar en iPhone y iPad. También existe **Android**, utilizado por muchos modelos de teléfonos inteligentes y tabletas, como la línea Galaxy de Samsung.

Naturalmente, para las computadoras tradicionales (sean de escritorio, *laptops* o *netbooks*) también existen opciones. Una de ellas es el tema de este artículo: Linux. Es un sistema operativo, por lo tanto también puede realizar todas las tareas que Windows te permite realizar.

Como lo comenté en mi artículo de la revista .Seguridad número 21, Linux es una excelente alternativa si tienes un equipo de cómputo con varios años de edad al que quisieras exprimirle aún más tiempo de vida útil. Esto es porque Linux es muy flexible. A diferencia de Windows que es producido por una única compañía (Microsoft), Linux es producido por cientos de diferentes organizaciones. Esta adaptabilidad permite que existan versiones de Linux (llamadas distribuciones o *distros*) con enfoques muy variados, incluyendo por supuesto *distros* específicamente diseñadas para equipos de bajo rendimiento. Ejemplos de esto incluyen Puppy, Arch y Lubuntu. Pero un equipo de bajo rendimiento no es la única



razón por la que pudieras querer utilizar Linux, pues este sistema operativo ofrece muchas ventajas: por ejemplo, es posible utilizarlo sin modificar tu equipo en absoluto, simplemente reiniciando con un CD (LiveCD) o memoria USB (LiveUSB). Linux suele ser 100% gratuito y eso incluye a los programas adicionales que quieras utilizar, esto lo hace una excelente opción para presupuestos limitados. Además es perfecto para servidores –muy útil para proyectos domésticos o personales–, como crear tu propia estación multimedia para tu casa o un servidor local de archivos y respaldos. También existen *distros* enfocadas a temas que pueden ser de tu interés, como la seguridad informática (Kali Linux).

Linux también es de interés para evitar la mayoría de las amenazas de *malware* que podemos llegar a encontrar debido a que en gran medida, éstas se desarrollan para Windows. Un ejemplo reciente es CTB Locker, que (de manera análoga a Cryptolocker el año pasado) "secuestra" los archivos del usuario cifrando el contenido hasta recibir un pago. Este *malware*, como la vasta mayoría de ellos, es completamente inocuo para usuarios de Linux.

Vale la pena resaltar que Linux no es inmune al *malware*; simplemente existe poco para infectar este sistema operativo. Esta situación puede cambiar en el futuro, sobre todo conforme aumenta la popularidad de esta plataforma. Es nuestra obligación estar informados y, como siempre, seguir las mejores prácticas de seguridad para estar apropiadamente protegidos en el futuro.

A continuación veremos algunos tips para escoger una distribución de acuerdo a tus necesidades, también te ofreceré algunos para poder migrar de Windows a Linux sin mucha dificultad.

Opciones, opciones...

Como escribí anteriormente, existen cientos de *distros* a tu disposición. Cada una es diferente, por supuesto, pero usualmente puedes seleccionar una con base en unos cuantos parámetros:

1. **Familia.** Prácticamente todas vienen de una de tres familias: Debian, Slackware y Red Hat. Estas son tres de las *distros* más antiguas que existen y las tres aún siguen en desarrollo activo (Red Hat bajo el nombre de Red Hat Enterprise Linux).

   a) La familia de Red Hat suele ser más popular en entornos corporativos, ya que Red Hat inició la tendencia de vender soporte técnico, facilitando su adopción. Sus descendientes más populares son RHEL, CentOS, Fedora y Mandrake.

   b) La familia de Slackware es la más antigua que sigue en desarrollo activo. Sus descendientes incluyen el propio Slackware, la familia de SuSE (SLES, openSUSE, etc.) y Gentoo Linux.

   c) La familia de Debian incluye a algunas de las *distros* más populares, como Ubuntu y Linux Mint.

La familia de Linux que escojas determina en varias maneras el alcance de tu equipo. Por ejemplo, la familia de Debian suele tener a las

más “amigables”; esta es una de las razones por las cuáles las tres *distros* más populares son de esta familia (Mint, Ubuntu y Debian), según el rastreo de DistroWatch.com. Esto a su vez, facilita la búsqueda de soluciones en Internet, ya que una mayor cantidad de usuarios en los foros de ayuda implica mayor tráfico y, por tanto, mayor atención.

Sin embargo, si estás explorando Linux como una alternativa en tu empresa o negocio, la familia de RedHat es probablemente la mejor opción. A distribuciones como RHEL y CentOS las distingue un servicio de soporte muy profesional, con altos niveles de servicio. Sueles encontrarlas en los servidores que algunas empresas venden con Linux preinstalado. Pero cuidado: su sesgo corporativo hace que algunas de estas *distros* estén disponibles en versión instalable a un costo (por ejemplo RHEL).

2. **Entorno gráfico.** A diferencia de Windows, Linux te permite escoger entre múltiples versiones de entornos gráficos. Los entornos más poderosos incluyen tecnología de punta como ventanas transparentes, efectos de transición y otros, pero requieren de equipos de cómputo más poderosos para poder funcionar apropiadamente. Muchas veces el entorno gráfico que elijas determinará en gran medida la cantidad de recursos que tu equipo de cómputo requiere, ya que el ambiente gráfico es uno de los principales consumidores de memoria y procesador en los sistemas operativos modernos.

Por ejemplo, para un equipo con pocos recursos probablemente prefieras utilizar Lubuntu, una variante del Ubuntu "clásico" que incluye al entorno gráfico LXDE. ¡Esto permite utilizarlo en equipos con hasta 128 MB de RAM!

En cambio, si deseas lo último en tecnología gráfica, puedes preferir la versión Cinnamon de Linux Mint, que requiere al menos 1 GB de RAM (aunque funciona mucho mejor con 2 GB).

3. **Arquitectura**. Linux está disponible para prácticamente todas las arquitecturas populares, y aunque seguramente puedes encontrar versiones de tu *distro* favorita para PC de 32 o 64 bits, posiblemente también las puedes encontrar para dispositivos tan variados como celulares, servidores SPARC e incluso dispositivos embebidos como la popular Raspberry Pi. Si piensas utilizar Linux para un proyecto que requiere el uso de una arquitectura específica, investiga primero si la *distro* de tu interés está disponible para la misma.

De cualquier manera, normalmente puedes descargar la distro de tu agrado desde su propio sitio.

LiveCD y LiveUSB: la alternativa práctica

Si aún no te decides a instalar Linux en tu equipo, deseas probarlo antes de realizar una instalación o simplemente quieres una excelente herramienta de emergencia para tu equipo, te recomiendo que utilices un LiveCD o (cada vez más popular) LiveUSB. El concepto en ambos es el mismo: un sistema operativo Linux completo, que puedes cargar en un CD o memoria USB y que puedes utilizar en cualquier computadora sin modificarla en lo absoluto.

Lo único que necesitas es la imagen ISO de la *distro* de tu interés y una herramienta para crear el LiveCD/USB. Para crear un LiveCD puedes utilizar prácticamente cualquier herramienta de “quemado” de discos; solo debes buscar la opción de crear a partir de una imagen ISO. Para USB yo utilizo la LinuxLive USB Creator (descarga gratuita) aunque existen muchas alternativas.

Puedes utilizar tu LiveCD/USB para particionar discos (por ejemplo con GParted), recuperar archivos borrados accidentalmente (con la utilidad forense foremost, por ejemplo) y muchas otras cosas más. Si trabajas en el sector de TI, un LiveCD/USB es una herramienta que te ayudará tremendamente en tus labores diarias. ¡Pruébala!

¿Y mis aplicaciones de Windows?

Aunque Linux tiene múltiples alternativas en su mayoría gratuitas y para distintas aplicaciones, es posible que necesites utilizar una aplicación que solo existe en Windows. Por ejemplo, tal vez requieres utilizar un filtro que solo está disponible en Photoshop o te es imperativo utilizar macros de Excel. Para estos casos puedes utilizar Wine o una versión comercial (de paga) del mismo con soporte técnico incluido, es llamado Crossover Linux.

Ambas te permiten la instalación de aplicaciones de Windows directamente en tu Linux. Puedes ejecutarlas como cualquier otra aplicación, según sea su nivel de compatibilidad, algo que puedes verificar fácilmente en el sitio web. Por ejemplo, Microsoft Office 2007 (que he utilizado para la creación de este documento) se ejecuta sin ningún problema en mi equipo Ubuntu.

Usualmente las aplicaciones con más problemas son los juegos, algo que debes tener en consideración si deseas migrar a Linux.

Finalmente, siempre existe la opción de una máquina virtual si cuentas con los recursos de cómputo suficientes. Una máquina virtual con Windows te consumirá como mínimo 1 GB de RAM (posiblemente 512 para XP) y usualmente 2 GB o más.

Si quieres saber más consulta:

- Recomendaciones de seguridad en una instalación Linux
- ¿Qué es el malware?
- Medidas preventivas para resguardar la información

Sergio Andrés Becerril

Es un profesionista informático con 15 años de experiencia en proyectos de la iniciativa privada, académicos y docentes. Actualmente labora en el Departamento de Cómputo del Centro de Enseñanza de Lenguas Extranjeras de la UNAM, y como Director de Tecnología de una consultora privada especializada en seguridad informática.

Puedes encontrarlo en twitter: @dolphone

Instalando un Sistema de Detección de Intrusos Inalámbrico (WIDS) en Raspbian (Funcionalidad) - II

José Luis Sevilla Rodríguez

En el artículo anterior describí los elementos técnicos necesarios para instalar y configurar OpenWIPS en Raspbian, detallé las características, funcionalidades y límites de la versión actual (versión 0.1 beta 1). Sin embargo, es importante enfatizar que la funcionalidad de la solución inalámbrica en este momento es únicamente de detección de ciertos ataques: desasociación de clientes específicos, desasociación de todos los clientes conectados al punto de acceso inalámbrico (AP) y fragmentación.

El objetivo de este artículo es mostrar cómo OpenWIPS realiza la detección de dichos ataques, describir en qué consisten, presentar y describir los beneficios de adoptar este mecanismo de seguridad en una organización.

Descripción de ataques

Ataque de desasociación. Es un ataque de denegación de servicio (DoS) que consiste en interrumpir la comunicación entre un AP y un cliente inalámbrico (computadora, laptop, celular, entre otros) o más. Existen utilidades como *aireplay-ng* que permiten mandar paquetes de desasociación.

Fragmentación. El estándar 802.11 especifica la fragmentación en la capa MAC (Control de Acceso al Medio), describe que un paquete de datos se divide en fragmentos cifrados de forma independiente y que cada uno utiliza el mismo *keystream* (flujo de caracteres pseudo-

aleatorios) al momento de transmitirlos. Después de interceptar un paquete de datos, el atacante es capaz de recuperar una porción de *keystream*.

Funcionalidad

Como primer paso después de la instalación de OpenWIPS se debe ejecutar el servidor tal como se muestra en la imagen 1.

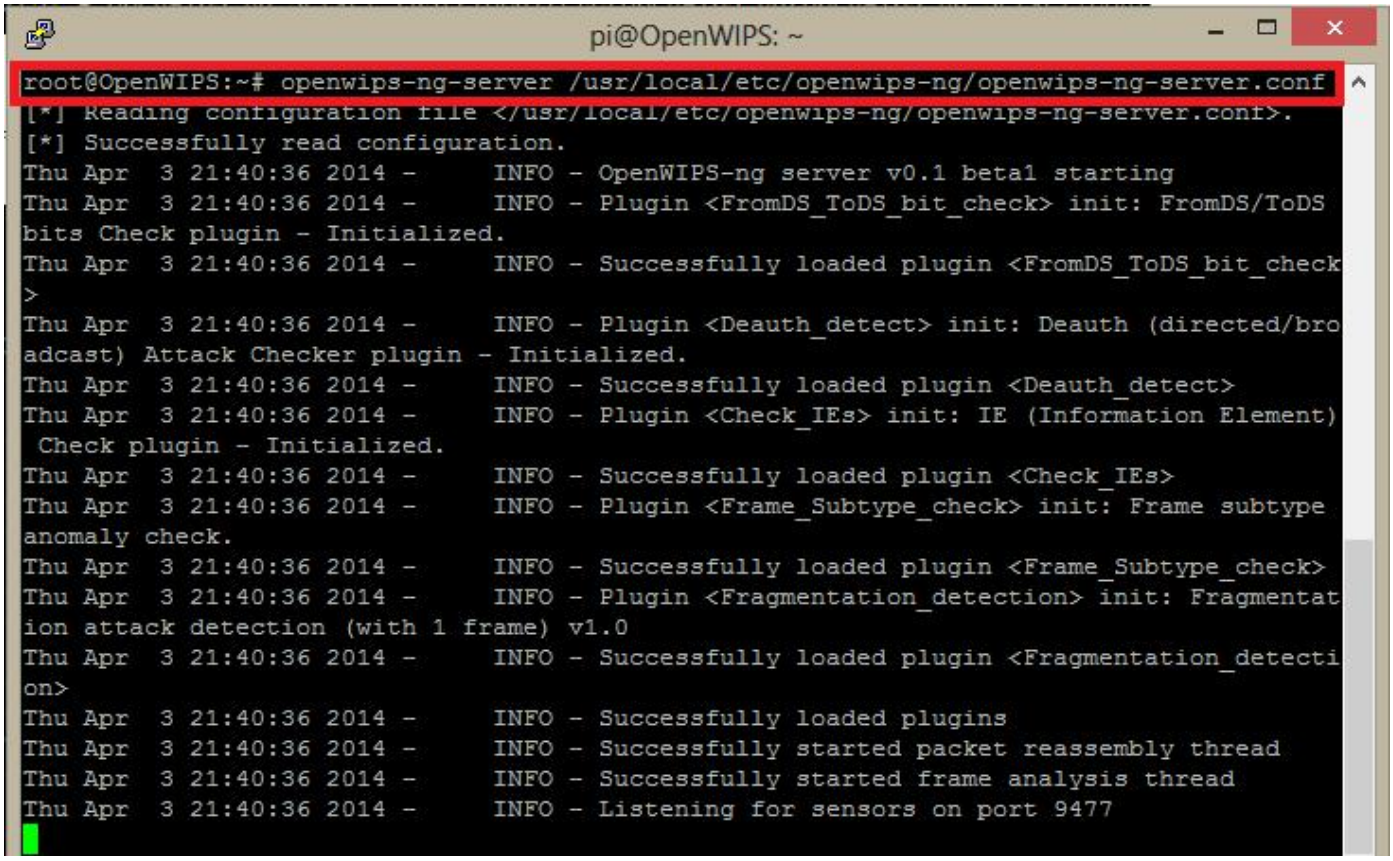


Imagen 1. Ejecución del servidor de OpenWIPS.

Después se debe conectar el sensor al servidor para que éste último procese el tráfico inalámbrico y pueda anunciar alertas (ver imagen 2).

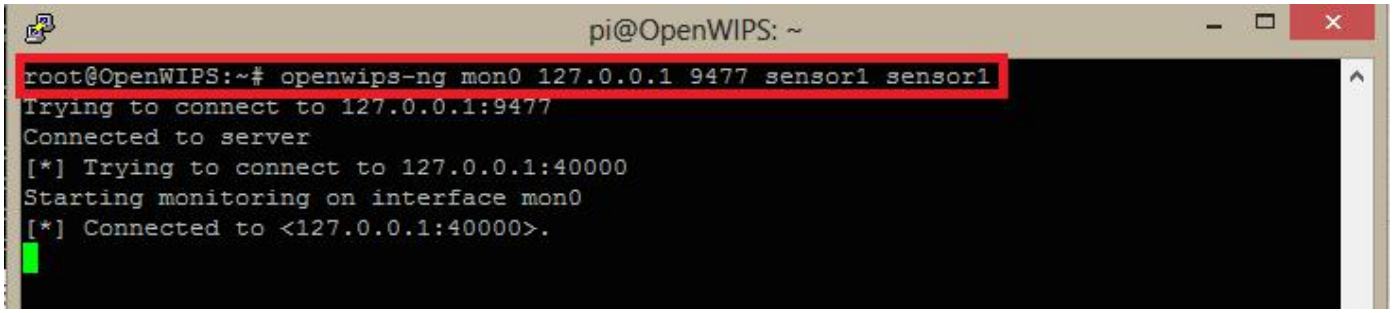


Imagen 2. Conexión del sensor al servidor.

Tal como se observa en la imagen 2, es importante que la tarjeta inalámbrica se encuentre en modo monitor (mon0) para captar todo el tráfico inalámbrico alcanzable y que éste pueda ser analizado por el servidor.

Para esta publicación, la dirección MAC del AP protegido es f4:55:9c:cc:fb:3c (la misma dirección MAC que en el artículo anterior) y el AP tiene asociados dos clientes 4c:eb:42:26:0e:5d y 50:32:75:ad:71:b7.

Si un atacante monitoriza el tráfico inalámbrico sería capaz de obtener el canal de transmisión (CH), la dirección MAC del AP (BSSID), tipo de seguridad (ENC), nombre de la red (ESSID), los clientes asociados al AP (STATION), entre otros (ver imagen 3).

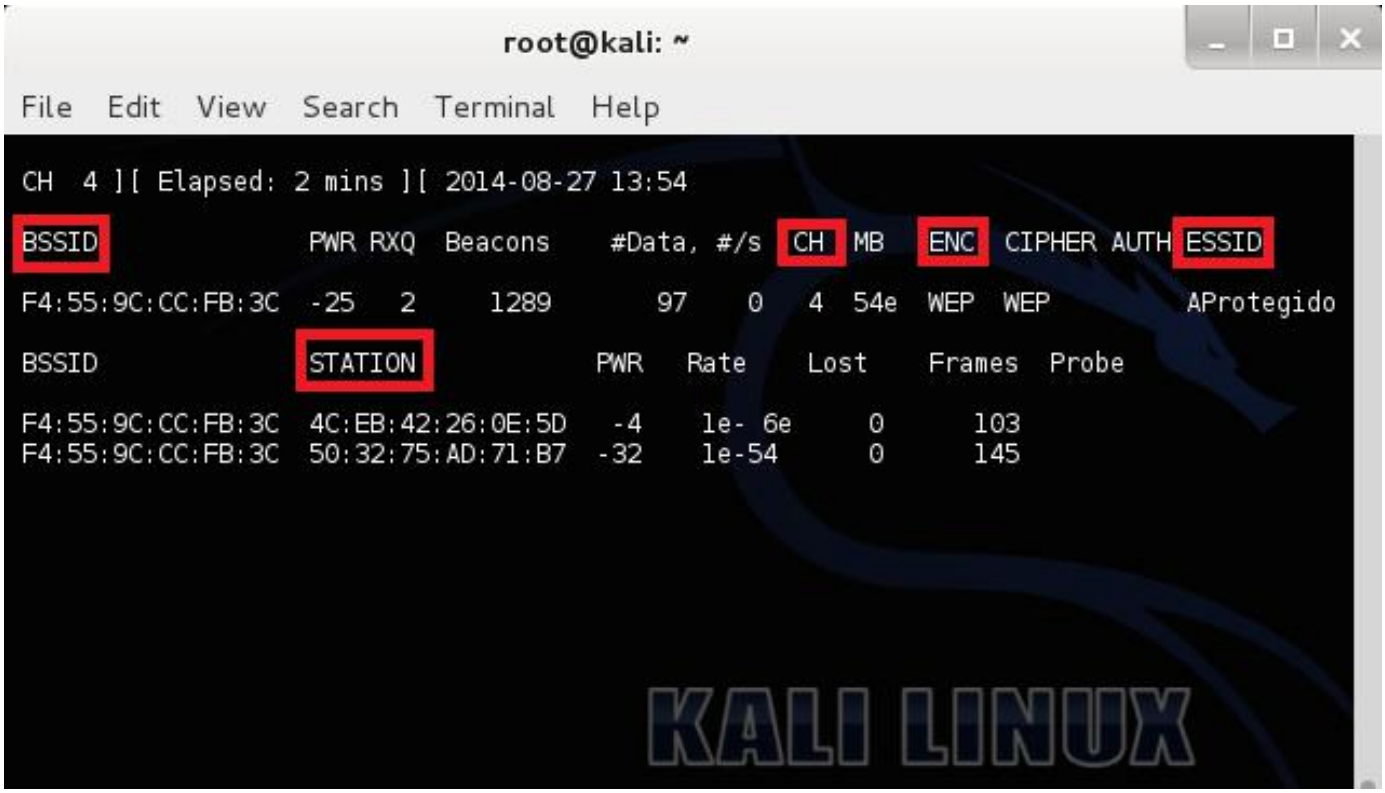


Imagen 3. Información que puede conseguir un atacante sobre un AP.

Supongamos que un atacante decide desasociar al cliente 50:32:75:ad:71:b7 que se encuentra conectado al AP (f4:55:9c:cc:fb:3c). En este caso quien sufrirá el ataque directamente es un cliente en específico. El objetivo de desasociar a un cliente en muchas ocasiones no solo es provocar un ataque DoS, si no capturar una clave precompartida entre el cliente y el AP.

Debido a que dicho AP está protegido por OpenWIPS, se dará alerta del ataque de denegación de servicio como se muestra en la imagen 4.

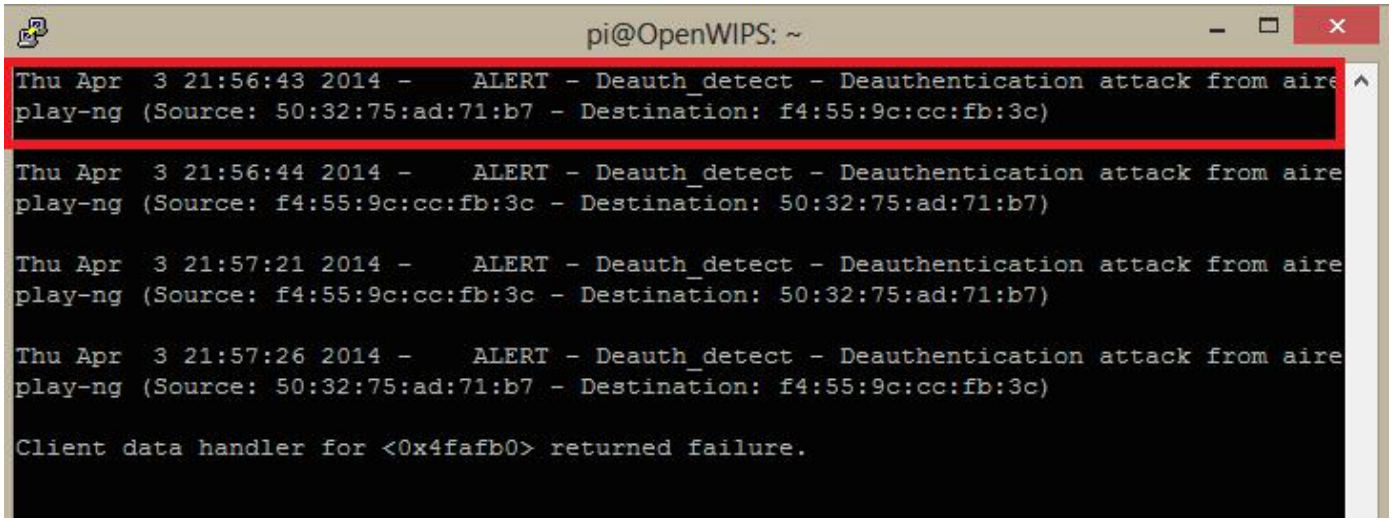


Imagen 4. Alerta de ataque de desasociación a un cliente.

Otro tipo de amenaza es un ataque de desasociación a todos los clientes conectados a un AP, en este caso se generaría una alerta tal como muestra en la imagen 5.

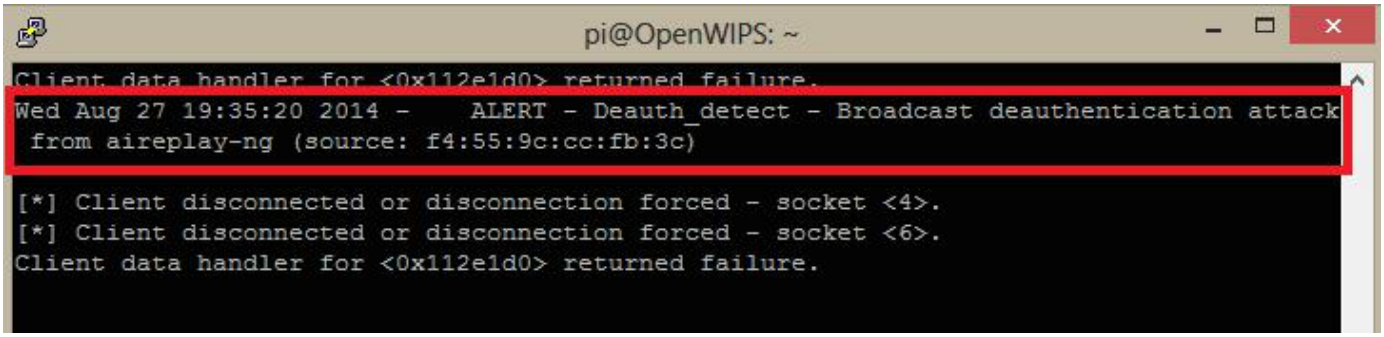


Imagen 5. Alerta de ataque de desasociación a todos los clientes conectados al AP.

En caso en que el atacante utilice técnicas o herramientas para fragmentación, OpenWIPS lo detectará de la siguiente forma (ver imagen 6).

Imagen 6. Alerta de ataque de fragmentación.

La dirección 00:c0:ca:57:b6:d3 corresponde al atacante; es importante tener en cuenta que la dirección física de una tarjeta inalámbrica puede ser modificada.

En las imágenes de alertas mostradas anteriormente existe información entre la que destaca la fecha y hora de la detección, el tipo de ataque y las direcciones MAC involucradas. Como mencionaba en la publicación anterior, esta información puede visualizarse en tiempo real mediante una interfaz y todas las alertas y eventos se guardan en una bitácora.

Beneficios de implementar OpenWIPS

Es importante resaltar que todas las organizaciones tienen necesidades diferentes, que dependen de su objetivo de negocio e infraestructura tecnológica, sin embargo, todas cuentan con información valiosa y por lo tanto es indispensable protegerla.

La implementación de OpenWIPS sobre Raspbian debe ser evaluada en caso de que se desee poner en operación y se debe analizar si es óptima para su infraestructura tecnológica. Los beneficios de esta solución son:

1. Fácil de implementar: la instalación y configuración es sencilla tal y como se documentó en el artículo anterior.
2. Código abierto: se pueden realizar mejoras o crear nuevas funcionalidades, por ejemplo, firmas de detección de otros ataques dirigidos a infraestructura inalámbrica, además, se pueden añadir módulos de IPS.

3. Bajo costo: una Raspberry Pi no es costosa (350-600 pesos mexicanos) y se necesita poco espacio físico.

4. Bitácora de evidencia: tener información sobre los incidentes ocurridos es una buena práctica para gestionar la seguridad de la información y, con base en los resultados obtenidos, se pueden tomar mejores decisiones.

5. Detección en tiempo real: los ataques son identificados en el momento y pueden ser tratados de acuerdo a los protocolos de reacción establecidos por la organización o, en caso de no tener procedimientos, proponer medidas de contingencia cuando se generen alertas de seguridad.

La solución sigue en desarrollo, sin embargo, se pueden realizar pruebas de concepto y comenzar la implementación por áreas pequeñas para después cubrir poco a poco la red inalámbrica de toda la organización.

Si quieres saber más consulta:

- El nuevo paradigma de seguridad en redes inalámbricas
- OUCH! Seguros en el camino
- OUCH! Protegiendo tu red casera

José Luis Sevilla Rodríguez

Egresado de la Facultad de Ingeniería de la carrera Ingeniería en Computación por la UNAM con módulo de salida en Redes y Seguridad. Cuenta con las certificaciones Ethical Hacker (CEH) y Hacking Forensisc Investigator (CHFI) de EC Council.

Formó parte de la tercera generación del programa de certificación de Cisco CCNA Exploration en Facultad de Ingeniería. Asimismo, egresó de la sexta generación del Programa de Becas de Formación en Seguridad Informática de UNAM-CERT. Laboró en el área de Auditoría y Nuevas Tecnologías de la CSI/UNAM-CERT, como especialista en pentest, auditor Interno. 2013 impartió los talleres Pruebas de Penetración en aplicaciones Web y Hacking ético.

Uno de los clásicos: *Buffer overflow*

Erika Gladys De León Guerrero, J. Inés Gervacio Gervacio

Buffer overflow es una de las vulnerabilidades persistentes a pesar de la evolución y complejidad de los mecanismos de seguridad. Se encuentra presente en diversas aplicaciones por lo que aparece constantemente en las listas de vulnerabilidades críticas publicadas por instituciones enfocadas a la notificación de nuevas amenazas de seguridad.

¿Qué es *buffer overflow*?

Buffer overflow es una vulnerabilidad causada por la inserción de datos con tamaño superior al esperado por una aplicación, lo que provoca la sobrescritura de espacios adyacentes en la memoria. Es común hacer una la analogía con un contenedor de agua (que representa el *buffer*) con capacidad para almacenar un litro (que representan los datos de la aplicación). Si en este recipiente se introducen tres litros, lo que ocurrirá es que un litro será adecuadamente vertido en el contenedor pero dos litros serán desbordados del recipiente. Esta es una descripción general de la vulnerabilidad, pero aquí pueden surgir diversas dudas, ¿qué es un *buffer*?, ¿qué pasa sí se sobrescriben espacios adyacentes de memoria?, ¿cuáles son las consecuencias de esta vulnerabilidad?

Para tener una concepción más adecuada de lo que es un *buffer*, es necesario recordar algunos conceptos sobre la memoria.

Estructura de la memoria

Cuando un programa es ejecutado, el sistema operativo reserva una zona de memoria para que la aplicación realice correctamente sus instrucciones, este espacio se divide en zonas de acuerdo a los distintos tipos de datos. Primero es necesario cargar el **código ejecutable** del programa, es decir, las instrucciones. La zona etiquetada como ***data*** es utilizada por lenguajes de programación que permiten la creación de

variables globales y variables estáticas.

Asimismo, se reservan por lo menos dos espacios para los datos requeridos en la ejecución, estos espacios son ***stack*** y ***heap***.

El ***stack*** almacena los argumentos de las funciones, las variables locales y las direcciones de retorno de las llamadas a funciones.

El ***heap*** se encarga de gestionar la memoria dinámica, es decir la memoria solicitada durante el tiempo de ejecución.

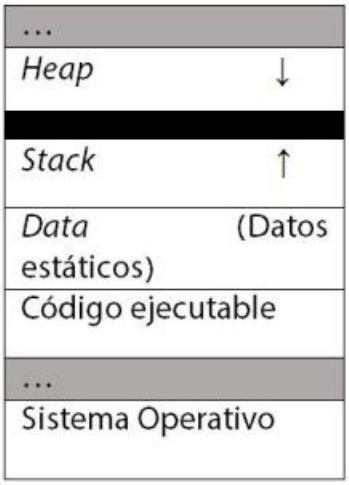


Imagen 1. Estructura de la memoria.

Existen regiones reservadas para datos de entrada del programa, pueden ser localizadas tanto en el *stack* como en el *heap* dependiendo del tipo de datos que van a almacenar. Estas regiones son llamadas ***buffer***, por lo que se puede definir el *buffer* como un espacio en memoria que sirve como almacenamiento temporal de datos de entrada en un programa.

Tipos de *buffer overflow*

Básicamente pueden distinguirse dos tipos primarios de *buffer overflow* que se ligan directamente con la explicación previa sobre las regiones de memoria ya que su nombre se deriva del espacio en memoria sobre el cual es localizada la vulnerabilidad:

- *Stack overflow*
- *Heap overflow*

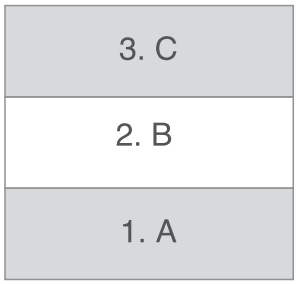
En este artículo se hablará sobre el tipo *stack overflow*.

Para comprender esta vulnerabilidad, es necesario conocer primero el funcionamiento del *stack*. La región de memoria reservada para el *stack* tiene una estructura de datos tipo LIFO (*Last In First Out*) donde el último dato que ingresa es el primero en salir. Existen dos operaciones que pueden realizarse en el *stack*, la operación "PUSH" para añadir elementos y "POP" para extraerlos, los datos almacenados se extraen uno a uno. Para entenderlo mejor puedes pensar en un conjunto de libros, colocando en primer lugar el de geometría (operación PUSH para cada uno), en segundo lugar el de cálculo y en tercer lugar el de álgebra. Para poder retirar el de cálculo, es necesario primero quitar el de álgebra (operación POP sobre el libro de álgebra) ya que fue el último en ser colocado.

Es posible describirlo de una forma más abstracta, introduciendo letras a un *stack*:

1. PUSH A
2. PUSH B
3. PUSH C

Inicio de *stack*



Fin de *stack*

Otro elemento importante para analizar un *buffer overflow* es la forma en la que se maneja una **llamada a procedimiento o función**. Al invocar una función se realiza un salto dentro de la memoria para ceder el control a las instrucciones que conforman dicha función. Una vez que termina de ejecutarse la llamada a procedimiento o función, es necesario que el programa sepa a dónde regresar para continuar ejecutando las siguientes instrucciones, por lo que es necesario almacenar un indicador en el *stack*, éste es

conocido como **dirección de retorno**, pieza clave para la ejecución de un *buffer overflow*.

Para entenderlo mejor, pensemos en un mesero al levantar una comanda en un restaurante, él atiende su mesa y pregunta por la orden, pasa esta orden a la cocina y ahí se realizan las instrucciones adecuadas para satisfacer el pedido, similar a como lo haría una función. Al terminar la preparación, es muy importante avisar al mesero que la orden está lista (dirección de retorno) y se le cede el control para que la haga llegar a la mesa.

Figura 2. Llamada a procedimiento o función.

Para observar cómo se manejan las variables y las llamadas a función, es posible recurrir al siguiente ejemplo, donde se inserta en el *stack* cada elemento del programa.

```
void funcion_vulnerable(int var1, var2, var3)
{
   char buffer1[5];
   char buffer2[10];
}

void main()
{
  funcion_vulnerable(a,b,c);
}
```

La memoria quedaría de la siguiente forma (tomando en cuenta los elementos descritos en la sección previa):

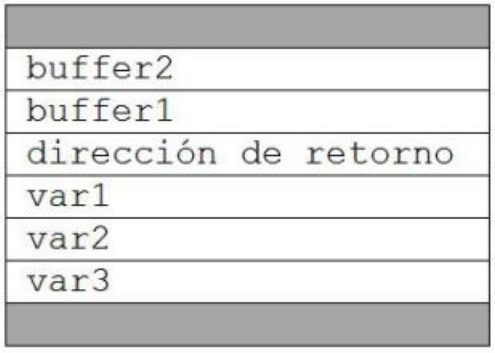


Figura 3. Estructura de stack.

Se realizan las operaciones PUSH sobre cada variable y al llegar a *funcion vulnerable*, se almacena en el *stack* la dirección de retorno para que el programa tenga la dirección a la cual deberá regresar cuando termine de ejecutar la función. Finalmente, una vez que se ejecutan las instrucciones de *funcion_vulnerable*, se almacena en el stack el contenido de cada *buffer*.

```
push var3
push var2
push var1
llamada a función  = introducir dirección de
retorno en stack
push buffer1
push buffer2
```

Figura 4. Operaciones sobre stack.

Nota: La descripción sobre la memoria es muy general y tiene como finalidad la comprensión del funcionamiento de la vulnerabilidad.

Finalmente, ya que se tiene una visión más clara sobre las operaciones realizadas sobre el *stack*, se puede ejemplificar un *stack overflow* con el siguiente código en lenguaje C.

El código valida con una condición, que se le haya pasado por lo menos un argumento al programa (*argc=argument count*) para llamar a "funcion" pasándole los parámetros insertados por el usuario (*argv=argument vector*) "cadena", la función genera un *buffer* "buffer1" de un tamaño de 8 caracteres y posteriormente copia "cadena" en "buffer1". El punto crítico se encuentra en esta instrucción, donde se copia la cadena enviada por el usuario en el *buffer*, finalmente se manda a imprimir en pantalla un texto.

```
#include <stdio.h>
void funcion(char *cadena);

int main(int argc, char *argv[])
{
   if(argc > 1)              //Validar paso de
parámetros del usuario
      funcion(argv[1]);      //Llama a funcion
y le envía parámetros del usuario

   printf("Entrada estándar menor o igual a 8
caracteres\n");
}

void funcion(char *cadena)
{
   char buffer1[8];          //Genera buffer
con espacio para 8 caracteres
   strcpy(buffer1, cadena); //Copia al buffer
la entrada de usuario
}
```

La ejecución del programa con el parámetro “AAAAAAAA” (8 caracteres), desplegará el siguiente resultado:

```
$ ./Codigo2 AAAAAAAA
Entrada estándar menor o igual a 8 caracteres
```

Todo se ejecuta de la manera esperada debido a que se han insertado en el *buffer* la cantidad de caracteres adecuados con los cuales se ha delimitado.

¿Qué pasaría si se ingresaran más caracteres de los que soporta el *buffer*, por ejemplo la cadena “AAAAAAAAA” (9 caracteres)?

```
$ ./Codigo2 AAAAAAAAA
*** stack smashing detected ***: ./Codigo2
terminated
…
0208c000-0504c000 rw-p 00000000 00:00 0
[heap]
…
decac000-d0000000 rw-p 00000000 00:00 0
[stack]

Aborted (core dumped)
```

La inclusión de una cantidad mayor a la permitida por los límites del *buffer* provoca un error. Se debe a que el *buffer* se ha desbordado y escribe los caracteres restantes en direcciones contiguas de memoria. Es ahí donde inicia el problema de un *stack overflow*.

Consecuencias del *stack overflow*

¿Qué pasa sí se sobrescribe la dirección de retorno? Se debe recordar que la dirección de retorno indica qué es lo que se debe ejecutar una vez que termina la ejecución de una función o procedimiento, en este caso, pueden ocurrir las siguientes opciones:

- Sí se llena con caracteres que no representen una dirección válida de memoria, generará un error de tipo “segmentation fault”. Por ejemplo, “AAAAAA” no es una dirección reconocible de memoria.
- Se puede apuntar a una dirección dentro del programa, provocando que se ejecute de forma incorrecta, por ejemplo, ir al inicio del programa o al final de éste, ir a una instrucción de impresión en pantalla, etc.
- Y el peor escenario y por el cual esta vulnerabilidad es muy explotada, es posible ejecutar código arbitrario:
  - Escrito por un individuo malicioso apuntando a una dirección en memoria donde inicie el código malicioso.
  - Código que ya se encuentra en el espacio de direcciones del sistema, haciendo referencia a la dirección donde se encuentre localizado, por ejemplo la ejecución de un Shell (/bin/sh).

Finalmente este artículo ha descrito un *buffer overflow*, en particular el de tipo *stack overflow*, se describieron los elementos principales para su comprensión y finalmente se resumieron sus consecuencias. Un punto importante para explotar esta vulnerabilidad, es encontrar la dirección de retorno y poder sobrescribirla.

Erika Gladys De León Guerrero

Maestra en Ingeniería en Seguridad y Tecnologías de la Información por el IPN, recibió el título de Ingeniera en Computación en la UNAM. Más de 10 años de experiencia en Seguridad de la Información, participando como desarrolladora, consultora, ponente, instructora y académica en diversos proyectos y cursos de gran importancia a nivel nacional e internacional.

J. Inés Gervacio Gervacio

Maestro en Administración de la Tecnología por la UNAM, Maestro en Ingeniería en Seguridad y Tecnologías de la Información por el IPN, Ingeniero en Computación por la UNAM. Colaboró por más de 10 años en UNAM-CERT donde participó en proyectos de Nuevas Tecnologías, Cumplimiento Regulatorio, Pruebas de Penetración, Auditorias de TI y Honeynet.

TIC (Internet) y ciberterrorismo

Alejandra Morán Espinosa, Abraham Alejandro Servín Caamaño, Oscar Alquicira Gálvez

La revolución tecnológica y la aparición de herramientas informáticas conocidas como Tecnologías de la Información y Comunicación (TIC) han caracterizado e impactado a las últimas décadas en prácticamente todas las ramas del conocimiento, han generado nuevas oportunidades, vínculos, negocios y hasta relaciones personales, facilitando la realización de actividades negativas como la existencia de conductas antisociales y de nuevos delitos.

En esta serie de artículos (tres entregas) se abordará desde la óptica del Derecho y de las Relaciones Internacionales el reto que representa el uso de estas nuevas herramientas para tres figuras del Derecho Internacional relacionadas con el conflicto y que hoy día son causa de polémica, nos referimos primero al **uso de la fuerza** (*ataques armados con la aplicación de la tecnología conocidos como ciberataques*), en segundo lugar al **espionaje** o *ciberespionaje* y finalmente al **terrorismo** o *ciberterrorismo*.

El ciberespacio ha generado, además de novedades, grandes dificultades legales principalmente por la falta de precisión territorial, sin embargo los principios del Derecho Internacional sí le son aplicables, por ello debe decirse que no es la primera vez que la tecnología logra modificar las herramientas tradicionales de un conflicto con el uso de las TIC pues el Derecho Internacional tiene que responder a esos nuevos panoramas, lo que provoca a su vez otros cambios en diversas ramas del mismo, como lo son el Derecho Humanitario Internacional y el Derecho de Guerra principalmente. Las reglas existentes del Derecho Internacional son también adjudicables a las conductas y los actos realizados a través del uso de estas nuevas tecnologías.

Por la importancia que revisten las TIC a esta temática, es pertinente abordar dentro de las mismas el uso específico de Internet. Para ello debemos comenzar por recordar el siguiente hecho: en 1957 la ex Unión Soviética puso en órbita el primer satélite artificial, el SPUTNIK, adelantándose así a los Estados Unidos en lo que se conoció como la carrera armamentista. Este hecho es importante porque marca el **comienzo de las comunicaciones globales**.

En respuesta, un año después el Departamento de Defensa de Estados Unidos creó la *Advanced Research Projects Agency* (ARPA por sus siglas en inglés) en conjunto con la NASA, la cual estaba encargada de realizar investigaciones con fines de protección ante las tenciones derivadas de la Guerra Fría y en específico de la posibilidad de un conflicto bélico.

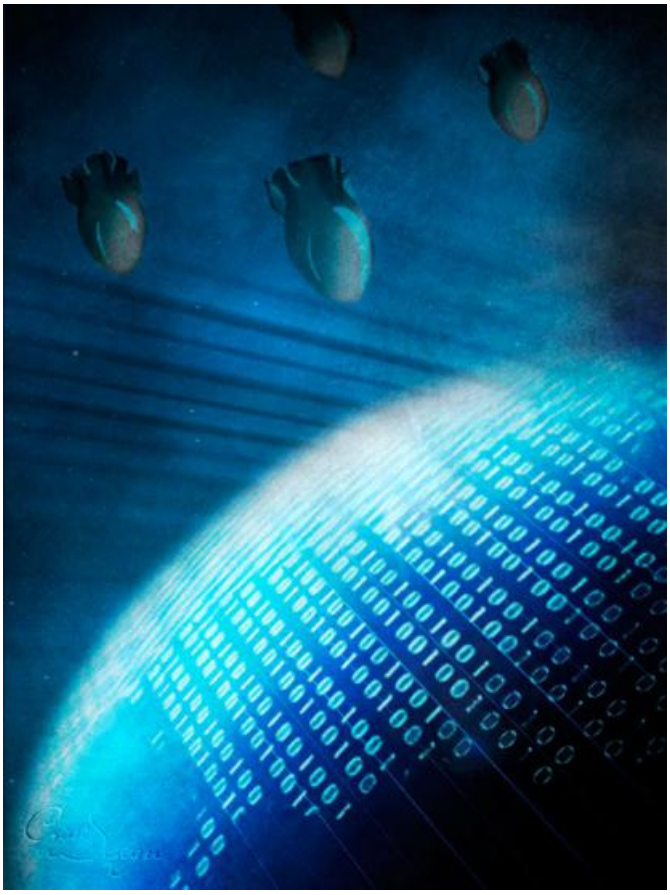
Uno de estos proyectos fue desarrollado e impulsado principalmente por Joseph Carl Robnett *Licklider*, quien en agosto de 1962 expuso su concepto de Red Galáctica (Galactic Network), donde concibió una red interconectada globalmente a través de la cual cada persona pudiera acceder desde cualquier lugar a la información y los programas. En 1969 se concretizaron una serie de trabajos que dieron como resultado la aparición de la primera *Red Académica* conocida como ARPANET, la cual tenía como característica el ser cerrada y de estricta índole académica, posteriormente se convirtió en DARPANET, que si bien también era una red cerrada, lo era en un menor nivel. En 1990 DARPANET se dejó de usar definitivamente y en 1991 el European Laboratory for Particle Physics in Switzerland (CERN) libera el World Wide Web o www desarrollado por Tim Berners-Lee.

Posteriormente se crearon los *browsers* (navegadores) que utilizaban HTTP, herramientas que debido a su naturaleza intuitiva y a la interfaz gráfica hacen que la www y la Internet sean más atractivas al público en general. Desde entonces Internet se ha desarrollado de manera inimaginable, en parte debido a los fines comerciales de las empresas. Internet ya no es la red de investigación ni militar para lo que fue creada, ahora Internet es, ante todo, un negocio, y eso ha sido lo que ha impulsado su desarrollo.[1]

El ciberespacio y el uso de la fuerza[2]

Derivado del uso de Internet, se hizo más frecuente la utilización indiscriminada del concepto *ciberespacio*, el cual Abril Cuarto (2010) considera como:

"Ámbito artificial o lugar virtual donde usuarios de la red interactúan a través de un lenguaje, expresado en sentido de textos, imágenes, gráficos, sonidos etc., entendiendo dicha red como un tejido de computadoras interconectadas que guardan bases de datos y fuentes de información, a las cuales los usuarios pueden acceder. Genera situaciones de derecho reales a pesar de romper el ámbito espacial, es decir, se puede caer en una determinada situación sin importar la distancia de las mismas, dicha situación sigue generando consecuencias de derecho a pesar de que no se realicen en un plano físicamente tangible." [3]



El ciberespacio, contrario a lo que pudiera pensarse, no es una zona totalmente libre de leyes donde cualquiera puede llevar a cabo todo tipo de actividades y conductas, incluyendo las hostiles. Las normas jurídicas fueron hechas para regular la conducta en sociedad y al igual que en la vida real (cotidiana y tangible), paulatinamente dichas normas pasarán a regular los diversos aspectos del mundo virtual.

Una persona que tiene una conducta hostil y utiliza una infraestructura tecnológica que se lo facilite genera necesariamente vínculos jurídicos al utilizar hardware, un servicio de conectividad, software y otras herramientas para evitar su localización. El resultado a corto o mediano plazo invariablemente llevará a que sea localizado y que solo baste identificar los principios jurídicos

acorde a las circunstancias, lugar, objeto y tipo de actividad para determinar el precepto jurídico aplicable y la jurisdicción que corresponda, lo cual es determinante para el momento de establecer el grado de responsabilidad, participación y penalidad que corresponda a la conducta realizada y el grado de daño causado.

Así, conflictos como los *ciberataques* a Estonia en 2007 que tuvieron como objetivo principal las redes financieras y comerciales de ese país, apagaron durante varios días su sistema bancario en línea. En el caso de Estonia, Internet es parte de la vida social del país a tal punto de que no solo se vota electrónicamente, incluso se pagan todos los impuestos y su banca *online* es una de las más desarrolladas.[4]

El ataque a Georgia en 2008, que inhabilitó el sitio web de la presidencia de este país además de otros sitios web gubernamentales durante el conflicto de su separación de Rusia, provocó que Georgia fuera incapaz de comunicarse vía Internet y, por lo tanto, no podía reubicar los activos norteamericanos.[5] En el caso de la República de Corea del 20 de marzo de 2013, los sistemas de los principales operadores de televisión y los bancos surcoreanos sufrieron un ciberataque a gran escala, lo que llevó a los militares a incrementar su nivel de alerta cibernética en medio de las preocupaciones sobre la República Democrática Popular de Corea, se atribuyó el ataque a este último beligerante.[6]

Los ciberataques mencionados son ejemplos de la incorporación del ciberespacio a distintas áreas del Derecho Internacional que evidencian el uso de la fuerza por medio de herramientas tecnológicas a través del ciberespacio, muestra clara de que **la guerra cibernética no es una ficción**.

Debe decirse al respecto que en ciertas circunstancias, las actividades en el ciberespacio (incluyendo el uso de la tecnología por supuesto), pueden ser consideradas como **uso de la fuerza nacional o internacional**, según el artículo 2 párrafo 4 de la Carta de las Naciones Unidas que manifiesta:

"Los Miembros de la Organización, en sus relaciones internacionales, se abstendrán de recurrir a la amenaza o ***al uso de la fuerza contra la integridad territorial o la independencia política de cualquier Estado****, o en cualquier otra forma incompatible con los Propósitos de las Naciones Unidas."*[7]

Dicha Carta también establece que **un Estado puede responder a un ataque por Internet** ejerciendo su derecho de legítima defensa en el caso de que el ataque sea equivalente a un ataque armado, ello con base en el artículo 51 del mismo documento que establece:

"Ninguna disposición de esta Carta menoscabará **el derecho inminente de legítima defensa, individual o colectiva, en caso de ataque armado contra un Miembro de las Naciones Unidas***, hasta tanto que el Consejo de Seguridad haya tomado las medidas necesarias para mantener la paz y la seguridad internacionales. Las medidas tomadas por los Miembros en ejercicio del derecho de legítima defensa serán comunicadas inmediatamente al Consejo de Seguridad, y no afectarán en manera alguna la autoridad y responsabilidad del Consejo conforme a la presente Carta para ejercer en cualquier momento* **la acción que estime necesaria con el fin de mantener o restablecer** *la paz y la seguridad internacionales."[8]*

La propia Carta de las Naciones Unidas usa el término **ataque armado** para describir los actos contra los cuales está permitido el uso de la fuerza a través del derecho de legítima defensa, sin

embargo, es innegable la relación entre los conceptos de agresión y ataque armado, es decir, por elemental inteligencia es claro que **todos los ataques armados se pueden equiparar o ser considerados una agresión directa**. Alexandrov Stanimir (1996), expresa sobre el término "*agresión*": "…***es cualquier uso ilegal de la fuerza, cualquier uso de fuerza que no sea legítima defensa contra un ataque armado o acción coercitiva por las Naciones Unidas***."[9]

A manera de conclusión de esta primera entrega, debemos entender entonces que un **ataque armado es causal de legítima defensa** y por tanto debe entenderse como la forma más seria y directa de agresión. Por consiguiente **no todos los tipos de agresión son causales de legítima defensa o no todos los casos de agresión son ataques armados**. Debe decirse que existen también las agresiones económicas e ideológicas como otros tipos de agresión pero que no usan al ciberespacio como medio.[10] Sin embargo, no existe limitación alguna para usarlo como tal. *De facto*, su propia naturaleza y características facilitan que sea un medio idóneo para realizar una agresión e incluso un ataque, aunque su armamento no sea de naturaleza bélica en principio.

En la próxima edición de la revista se retomará con más detalle lo anteriormente explicado con un enfoque de aplicación al ciberespacio y también se hablará del ciberespionaje y del ciberterrorismo.

Si quieres saber más consulta:

- Investigadores militares publican manual para ciberguerra
- Acuerdos Internacionales para la Privacidad de la Información

Referencias

[1] Trejo Garcia, Elma del Carmen (2006), "Regulación Jurídica de Internet", Cámara de Diputados, Centro de Documentación, Información y Análisis, Dirección de Servicios de Investigación y Análisis, Subdirección de Política Exterior, México.

[2] Para la siguiente parte se tomaron en cuenta dos fuentes principales: La posición oficial del asesor jurídico del Departamento de Estado de los Estados Unidos Harol Hogju Koh plasmada en la página de dicha oficina y el discurso de Harol Hogju Koh con un grupo de expertos en Derecho Internacional que dieron a conocer el Manual Tallinn en el Centro de Excelencia en la Defensa Cooperativa Cibernética de la OTAN. El manual toma las reglas vigentes sobre el comportamiento en el campo de batalla, por lo que se destaca la contraposición de algunos puntos con el discurso de Koh.

[3] Abril Cuarto, Gonzalo. El cuarto bios. Estudios sobre comunicación e información. Editorial Complutense, Madrid, 2010. pág.83-89.

[4] Lander, Mark y Markoff, Mark, Digital fears Emerges After Data Siege in Estonia, The New York Times. Consultado en abril 21 de 2013. Disponible en: http://www.nytimes.com/2007/05/29/technology/29estonia.html?_r=2&hp&oref=slogin&

[5] Markoff, John, Before Gunfire, Cyberattacks, The New York Times. (Consultado en abril 21 de 2013). Disponible en: http://www.nytimes.com/2008/08/13/technology/13cyber.html

[6] Kwon, KJ y Mullen, Jethro. Ataque cibernético a bancos y televisoras paraliza Corea del Sur. CNN. Consultado en mayo 6 del 2013. Disponible en: http://cnnespanol.cnn.com/2013/03/20/ataque-cibernetico-a-bancos-y-televisoras-paralizan-corea-del-sur/

[7] Organización de las Naciones Unidas. Carta de las Naciones Unidas. Consultado en marzo 30 del 2013, disponible en: http://www.un.org/es/documents/charter/).

[8] Carta de las Naciones Unidas. Op.cit.

[9] Stanimir, A. Alexandrov, Self-Defense Against the Use of Force in International Law. The Netherlands, Kluewer Law International. 1996. Pág. 108. (Opinión derivada de las propuestas de la República Dominicana, México, Paraguay y Perú durante las sesiones de la Comisión de Derecho Internacional de 1951).

[10] Ibídem. pág. 105-114.

Alejandra Morán Espinosa

Licenciada en Derecho por la UNAM con mención honorífica, candidata a Maestra en Política Criminal, especialista en Derecho Informático y nuevas tecnologías, profesor de Derecho Informático en la FES Acatlán y universidades privadas, ponente en temas de delitos informáticos, ciberseguridad y tecnologías de la información y comunicación.

Abraham Alejandro Servín Caamaño

Licenciado en Relaciones Internacionales por la UNAM, profesor adjunto de las materias de seminario de política exterior, derecho internacional público y derecho marítimo en la FES Acatlán, cursante de la maestría Maritime Law en la Universidad de Southampton en Reino Unido.

Oscar Alquicira Gálvez

Licenciado en Derecho, Profesor adjunto de derecho informático en la FES Acatlán y colaborador en investigación jurídica y de nuevas tecnologías en el laboratorio IUSTICS en FES Acatlán 2011-2013 apoyando en la investigación y realización de contenidos en línea para la enseñanza del campo jurídico a Distancia.

Riesgos de seguridad en Android

Miguel Ángel Mendoza López

La consolidación de Android como el sistema operativo más utilizado en los dispositivos móviles de la actualidad lo ha convertido en blanco de diversas amenazas que continúan creciendo, una tendencia que se ha visto en los últimos años.

De acuerdo con IDC, hasta el tercer trimestre de 2014, Android ocupaba el 84.4% del mercado de dispositivos móviles. Actualmente se estima que el número de *smartphones* y tabletas que tienen instalado este sistema operativo, supera los mil millones y que su uso seguirá en aumento, ya que ahora será posible encontrarlo en otros dispositivos inteligentes como relojes, televisores e incluso en automóviles.

De forma paralela, la cantidad y diversidad de la información que los usuarios almacenan, procesan o transmiten a través de estos dispositivos, cuando utilizan servicios de Internet como correo electrónico, redes sociales o banca en línea, contribuye a aumentar su importancia y en consecuencia, el interés de los cibercriminales por la plataforma móvil. Como resultado, en los últimos años se ha observado un incremento en el desarrollo de software malicioso que tiene como objetivo este popular sistema operativo.

Aunado al *malware*, en este artículo revisaremos otras amenazas enfocadas en Android, con el propósito de conocer los riesgos a los que está expuesto un usuario cuando utiliza esta tecnología. De esta manera, damos el primer paso en el camino para proteger nuestra información.

Malware desarrollado para Android

Una de las amenazas más relevantes es el software malicioso desarrollado de forma específica para esta plataforma. La cantidad, complejidad y diversidad del *malware* que se genera para Android ha crecido de manera

considerable en los últimos años.

Nuevas familias de códigos maliciosos y sus variantes se desarrollan con distintos objetivos. Entre estos se encuentran troyanos SMS que suscriben al usuario a servicios de mensajería *premium* sin su consentimiento, *botnets* que buscan convertir en zombi al dispositivo móvil al tiempo que roban información, *adware* para el envío de publicidad no deseada o *ransomware* que cifra la información y solicita un pago como rescate para que el usuario pueda recuperar sus datos.

De acuerdo con el documento "Tendencias 2014: el desafío de la privacidad en Internet" de ESET Latinoamérica, en donde se estudiaron los registros de detección de software malicioso, en el año 2010 se identificaron solo tres familias de *malware* para Android, para 2011 fueron 51, a finales de 2012 se detectaron 63, mientras que en 2013 ya se habían identificado 79 familias.

Del mismo modo, se comenzaron a reportar subcategorías de troyanos que solo se identificaban en plataformas como Windows, tal es el caso de *downloaders* (que permiten descargar otras amenazas desde Internet para instalarlas posteriormente), *droppers* (que instalan otras amenazas que están incluidas dentro de este programa), *clickers* (para generar tráfico hacia sitios o avisos publicitarios) o bancarios (para obtener datos relacionados con entidades financieras).

Por ejemplo, el año pasado se identificó el primer caso de *malware* de la familia *filecoder*, un troyano (Simplocker) que una vez dentro del dispositivo, se encarga de buscar archivos como fotos, videos o documentos, luego los cifra y exige un pago para que el atacante pueda proporcionar la clave que permite descifrar la información del usuario.

Además, las variantes de cada familia han aumentado también, principalmente cuando se realizan ligeras modificaciones a un código malicioso conocido, con la intención de evadir la detección de las soluciones de seguridad o para agregar nuevas funcionalidades maliciosas.

Fragmentación de Android

Se trata de un importante problema en este sistema y se refiere al hecho de que no todos los dispositivos que funcionan con Android utilizan la misma versión, o al menos la más reciente. En ocasiones, las actualizaciones del sistema son retenidas por los fabricantes de los dispositivos para evitar problemas de compatibilidad, lo que evita que las correcciones de seguridad lleguen a los usuarios.

En otros casos se lanzan al mercado nuevos modelos de *smartphones* con versiones anteriores de la plataforma, debido principalmente a que se trata de desarrollos específicos para un tipo de dispositivo móvil.

Se han puesto en marcha distintas iniciativas que tienen como objetivo consolidar las versiones de Android de manera que la mayoría de los usuarios puedan hacer uso de la última versión, sin embargo, esto todavía no se logra. Como consecuencia, utilizar versiones desactualizadas se traduce en un importante riesgo de seguridad.

Con base en información publicada en el tablero

de control de Android (*dashboard*) para las versiones de la plataforma, cada mes, durante un periodo de siete días, se actualizan los porcentajes de distribución de las distintas versiones utilizadas a partir del número de dispositivos que acceden a Google Play.

En esta recolección de datos que concluyó el 2 de marzo de 2015, la versión 5.0 (Lollipop) ocupa el 3.3%, mientras que la versión 4.4 (KitKat) alcanza 40.9% del total de dispositivos. Jelly Bean 4.1.x aparece con 17.3%, la versión 4.2.x con el 19.4% y la 4.3 con 5.9% de la distribución. En el informe todavía se puede encontrar la versión 2.2 (Froyo), que se mantiene en 0.4% del total de los equipos con Android instalado.

Sistemas operativos personalizados

Otro riesgo de seguridad en Android está relacionado con la modificación del sistema operativo cuando se personaliza. Los fabricantes de dispositivos o los operadores de telefonía móvil (*carriers*) modifican el sistema para agregar información de su compañía, incluso los usuarios también pueden realizar cambios en el aspecto de su teléfono a través de las capas de personalización o lanzadores (*launchers*).

Esta es una de las características por la cual algunas personas prefieren Android sobre otros sistemas operativos, debido a la flexibilidad para la modificación. Sin embargo, los cambios realizados pueden generar huecos de seguridad, ya que se realizan modificaciones, en muchas ocasiones sin saber lo que realmente sucede en el teléfono.

En una investigación realizada por FireEye, se reportó un ataque de *phishing* derivado de permisos ligados a la configuración de un lanzador de Android. Los investigadores identificaron que la modificación en los iconos de la pantalla principal del dispositivo no solo cambiaron de apariencia, sino que además dejaron de apuntar hacia las aplicaciones que deberían y en su lugar redirigían al usuario a una página web maliciosa.

Apps en repositorios de terceros

Una de las razones por la cual se ha extendido el uso de Android es la cantidad de aplicaciones desarrolladas para esta plataforma, lo que permite tener nuevas funcionalidades en el dispositivo. En ocasiones, con la idea de obtener de manera gratuita una aplicación de paga, los usuarios pueden descargarla desde sitios no oficiales, en donde es posible que una *app* haya sido modificada para realizar acciones maliciosas.

Por ejemplo, en una investigación realizada por el laboratorio de ESET Latinoamérica, se identificaron aplicaciones para móviles (principalmente juegos) que propagaban *Boxer*, un troyano SMS que afectó a 63 países, incluidos 9 de Latinoamérica. La aplicación podía descargarse de repositorios no oficiales (como un archivo apk) e instalarse en el teléfono.

Luego de que el usuario instala y juega con la aplicación, el programa maliciosos identifica el Código Móvil de País (*Mobile Country Code*) y el Código Móvil de Red (*Mobile Network Code*), datos numéricos utilizados de manera conjunta para conocer el país y los operadores de telefonía móvil, posteriormente envía mensajes que suscriben a servicios de mensajería SMS premium.

El usuario percibe las anomalías en su teléfono cuando debe pagar las cuentas por los servicios contratados sin su consentimiento, ya que se bloquean las notificaciones de contratación que las operadoras de mensajería envían a sus clientes.

Filtración de apps maliciosas en el repositorio oficial

Para la publicación de aplicaciones en Google Play es necesario registrar información de los desarrolladores y de la *app* antes de comercializarla, entre otras medidas, como la firma de las aplicaciones. A pesar de los esfuerzos por mantener la tienda libre de *malware* (por ejemplo con el desarrollo de Google Bouncer), existe el riesgo de que aplicaciones maliciosas puedan filtrarse en el repositorio oficial.

Cuando esto sucede, se verifica que se trata de una aplicación que contiene *malware* y se elimina de la tienda. Sin embargo, para este momento puede ser tarde, ya que al estar disponible en el repositorio oficial, puede ser descargada por los usuarios. En estos casos, el impacto depende del número de usuarios que hayan instalado la aplicación maliciosa en su dispositivo.

Esto sucedió con aplicaciones maliciosas encontradas en Google Play, que propagaban una amenaza denominada DroidDream, un código malicioso que operaba cuando el usuario no realizaba actividades con su *smartphone*, uniéndolo a una *botnet*. Se estima que afectó a 250,000 equipos, con base en el número de descargas que tuvieron estas apps.

Debido a su impacto, Google tuvo que llevar a cabo un procedimiento (denominado *kill switch*) para desinstalar de manera remota las aplicaciones maliciosas de los dispositivos infectados. Como último recurso y sin el consentimiento de los usuarios, se obtuvo acceso a los teléfonos móviles para remover las aplicaciones que contenían *malware*.

Recientemente se identificaron otras aplicaciones para *Android* del tipo RAT (troyanos de acceso remoto) que siguiendo el mismo principio, se hacían pasar por aplicaciones legítimas, como *apps* para llevar a cabo transacciones bancarias, monitorear el uso de datos, incluso antivirus para dispositivos móviles. Entre sus principales actividades se encontraba la recopilación de información del dispositivo infectado, conexiones a un servidor, descarga y ejecución de módulos con otras actividades maliciosas.

Vulnerabilidades en el software

Las vulnerabilidades que pueden presentarse para los dispositivos móviles son variadas, ya que pueden incluir debilidades en el sistema operativo, en las aplicaciones o en software que no es propio de la plataforma, pero que lo afectan de forma directa, como es el caso de *Heartbleed*, la falla en *OpenSSL*.

En cuanto al sistema operativo *Android*, generalmente las vulnerabilidades afectan de manera particular solo a algunas versiones del sistema y pueden ir desde permitir el acceso a la memoria física del dispositivo, hasta otorgar privilegios de administrador sobre el mismo.

En el caso de las aplicaciones, éstas también pueden incluir vulnerabilidades, como sucedió con *InMobi*, una aplicación utilizada para insertar publicidad en otras *apps*. Si la vulnerabilidad era explotada, un atacante podía enviar mensajes SMS o tener acceso a información sensible del usuario. El impacto es mayor debido a que otras aplicaciones utilizan esta herramienta para mostrar publicidad en sus propios desarrollos.

Por otro lado, con la aparición de *HeartBleed* se dio a conocer que la versión 4.1.1 (*Jelly Bean*) de Android estaba en riesgo debido a que utiliza una versión vulnerable de OpenSSL. Google emitió un parche de seguridad para esta versión en busca de reducir el problema. El inconveniente es que algunos equipos no soportarían la migración a una versión superior.

Entonces, ¿cómo protegerse de los riesgos?

Hemos enlistado estas amenazas junto con un recuento de sucesos para conocer los riesgos

de seguridad a los que está expuesto un usuario de Android cuando utiliza su dispositivo móvil, ya que estar consciente de ellos es el primer paso para evitar o reducir sus consecuencias. Para proteger la información, es necesario conocer las características del dispositivo y configurarlo de acuerdo a las preferencias y necesidades, evitando en la medida de lo posible tener las opciones predeterminadas.

Es importante recordar que cuando algo suena demasiado bueno para ser cierto, es muy probable que sea malicioso, esto evitará que seas víctima de algún ataque de ingeniería social, ya que una cantidad importante de *malware* se distribuye a través de juegos recompilados, falsas soluciones de seguridad o aplicaciones con distintas utilidades que aparentan ser inofensivas. Además, se está convirtiendo en una necesidad contar con herramientas de seguridad en el dispositivo.

Finalmente, llevar a cabo una serie de buenas prácticas en el uso de los teléfonos inteligentes incluye, entre otras, la instalación y actualización de software contra códigos maliciosos, verificación de los permisos solicitados por las aplicaciones que se instalan en el teléfono, uso de cifrado, respaldo de información, actualización del sistema operativo o el uso de las últimas versiones de las aplicaciones firmadas por los desarrolladores. Estas medidas nos permitirán disfrutar de la tecnología en un ambiente más seguro.

Si quieres saber más consulta:

- 10 consejos para mantener nuestra seguridad en el celular
- Uso seguro de aplicaciones móviles
- Análisis de *malware* móvil en Android

Referencias

Guía de seguridad para smartphones: ¿Cómo configurar tu Android de la forma más segura? ESET Latinoamérica. Junio de 2014.

Tendencias 2014: El desafío de la privacidad en Internet. ESET Latinoamérica. Noviembre de 2013. http://www.welivesecurity.com/wpcontent/uploads/2014/01/troyano_SMS_Boxer.pdf. Consultada al 10 de enero de 2015.

http://www.fireeye.com/blog/technical/2014/04/occupy_your_icons_silently_on_android.html. Consultada al 24 de septiembre de 2014.

https://developer.android.com/about/dashboards/index.html. Consultada el 3 de marzo de 2015.

http://www.welivesecurity.com/la-es/2011/03/09/google-vs-droiddream/. Consultada al 24 de septiembre de 2014.

http://www.welivesecurity.com/la-es/2014/04/14/usuarios-dispositivos-moviles-tambien-victimas-heartbleed/. Consultada al 24 de septiembre de 2014.

Miguel Ángel Mendoza López

Ingeniero en Computación por la Facultad de Ingeniería de la UNAM, Miguel Ángel Mendoza se desempeña actualmente como Security Researcher en ESET Latinoamérica, compañía dedicada al desarrollo, investigación y comercialización de soluciones de protección antivirus y seguridad informática. Además se desempeña como vocero de ESET Latinoamérica y representa a la empresa en todo tipo de actividades tales como seminarios, conferencias, capacitaciones internas y otros eventos de exposición pública.

Colaboró en la DGCCH de la UNAM, en la Facultad de Ingeniería y formó parte de la Coordinación de Seguridad de la Información/UNAM-CERT en el área de Auditoría y nuevas tecnologías, donde desarrolló actividades de implementación de estándares, mejores prácticas y auditorías de seguridad informática.

DGTIC

DIRECCIÓN GENERAL DE CÓMPUTO Y DE
TECNOLOGÍAS DE INFORMACIÓN Y COMUNICACIÓN

UNAM
CERT

Revista .Seguridad Cultura de prevención para TI
No.23 / marzo-abril 2015 ISSN: 1251478, 1251477