Implementación de esquema de seguridad a bajo costo para servicios en sistemas operativos GNU/Linux usando *uncomplicated firewall* y *secure shell tunneling*

Luis Alberto López González
Instituto Tecnológico de Celaya
luislao@itcelaya.edu.mx

Juan Ignacio Cerca Vázquez
Instituto Tecnológico de Celaya
nacho@itcelaya.edu.mx

José Jesús Sánchez Farías
Instituto Tecnológico de Celaya
jesus.sanchez@itcelaya.edu.mx

Oscar Grimaldo Aguayo
Instituto Tecnológico de Celaya
oscar.grimaldo@itcelaya.edu.mx

José Guillermo Rodríguez Villafaña
Instituto Tecnológico de Celaya
guillermo.rodriguez@itcelaya.edu.mx

Resumen

En una institución u organización los servidores proveen una serie de servicios a usuarios internos y externos, siendo capaces de almacenar información considerada como sensible o incluso confidencial. Comúnmente estos servidores pueden proveer páginas web, correo electrónico, autenticación, base de datos, almacenamiento de archivos, protección a la red, resolución de nombres de dominio, etc.

Debido a esto los servidores son frecuentemente el objetivo de usuarios maliciosos, por el valor de la información contenido en ellos. Por ejemplo un servidor que contiene un Sistema Gestor de Base de Datos puede ser atacado para lograr el acceso a los datos almacenados.

Las áreas de Tecnologías de Información (TI) de las organizaciones deben de ser cuidadosas y planificar los aspectos de seguridad concernientes a sus nuevos servidores. Tener políticas, estándares y procedimientos de seguridad en los aspectos de TI es una buena práctica para mantener la integridad y disponibilidad de la información.

Herramientas libres como *Uncomplicated Firewall* (UFW) y *OpenSSH server* nos pueden apoyar tecnológicamente en mantener nuestros servicios seguros.

Introducción

Las organizaciones por ley están obligadas a mantener los datos confidenciales (Ley Federal de Protección de Datos Personales en Posesión de los Particulares, 2010) debiendo haber considerado los aspectos de seguridad en cuanto a temas tecnológicos se refiere.

Deben asegurarse que cada servidor instalado en la red este configurado para satisfacer los requerimientos de seguridad que las leyes exigen. Configurar adecuadamente los aplicaciones que darán algún servicio a la red puede reducir considerablemente el riesgo de que personas no autorizadas tengan acceso a la información almacenada en el servidor o transmitida entre el cliente-servidor.

Implantar medidas de seguridad en servidores GNU/Linux nos puede llevar a alguna de las siguientes tareas: configurar, actualizar y parchar el sistema operativo o las aplicaciones, desinstalar servicios innecesarios, establecer controles de protección, configurar métodos de autenticación y de privilegios y llevar a cabo pruebas de seguridad.

Bloqueando servicios sin perder el acceso

Un punto importante de partida para planificar las políticas de seguridad es conocer los riesgos asociados a las aplicaciones/servicios que tenemos instalados en nuestros servidores.

Un servidor es un equipo que provee uno o más servicios a otros equipos en la red. Por ejemplo un servidor de archivos provee a los usuarios de la red la posibilidad de acceder, modificar, almacenar y eliminar los documentos almacenados en él. Un servidor de base de datos provee

los mecanismos para que los usuarios, accedan, manipulen y preserven la información almacenadas en las distintas tablas.

Por ejemplo, el Sistema Gestor de Bases de Datos (SGBD) PostgreSQL provee mecanismos para que los usuarios sean autenticados y para que accedan a la información contenida en las bases de datos o tablas donde se tengan los privilegios de lectura, inserción, actualización o eliminación. Así como el mecanismo para determinar que host tiene acceso a que base de datos y a través de cuál dirección IP o rango se llevará a cabo la conexión. La autenticación de los clientes es controlada por un archivo de configuración, comúnmente llamado pg_hba.conf (PostgreSQL Documentation, 2014), el formato general de esté archivo es mediante una lista de registros, cada registro se encuentra en una línea y específica el tipo de conexión que se llevará a cabo (ej. local o por host), si es por host se deberá especificar la IP o rango de IP, así como el nombre de la base de datos a la cuál se accederá y el método de autenticación que se usará (ej. peer, md5, trust, ssl).

Cada registro posee la siguiente estructura:

```
Host|local database  user [ address ] auth-method
```

Para intentar conectarse a una base de datos particular cada usuario debe tener una línea especificada en este archivo, si no la tiene automáticamente la conexión será rechazada. Si es exitoso entonces la base de datos determinará a nivel SGBD si el usuario tiene el privilegio de CONNECT/LOGIN.

Sin embargo mientras más usuarios y ubicaciones se conecten al SGBD más líneas debemos configurar en archivo pg_hba.conf. Resultando en la posible pérdida del control del mismo.

Un mecanismo que se puede implementar es no modificar los archivos de configuración que predeterminadamente sólo garantizan el acceso a las bases de datos de manera local, es decir únicamente el servidor que contiene la base de datos puede acceder a ella, ningún otro equipo en la red puede.

En primera instancia esto no tiene sentido ya que el propósito del servidor es proveer servicios a red, y se está sugiriendo bloquearlo.

Sin embargo podemos implementar un esquema de seguridad para que aún bloqueando los accesos predeterminados por PostgreSQL nuestros clientes puedan acceder a la base de datos.

Ejemplo:
Instalaremos *uncomplicated firewall* (UFW) para bloquear el puerto 5432, qué es el que utiliza PostgreSQL para escuchar las peticiones.

```
# apt-get install ufw
# ufw default deny incoming
# ufw deny 5432/tcp
```

De está manera ningún cliente en la red podrá acceder a la base de datos, sin importar que tenga o no una línea en el archivo pg_hba.conf.

¿Cómo podemos acceder?

Un mecanismo para acceder a la base de datos, aún estando bloqueado a nivel de firewall es a través de un túnel SSH.
Un túnel ssh crea una conexión segura entre una computadora local y una máquina remota (SSH/OpenSSH/PortForwarding, 2014), pudiendo especificar que servicios pueden ser redirigidos. Otro punto importante es que está tecnología garantiza que la conexión entre los dos puntos este encriptada.
Antes de crear el túnel se deben permitir las conexiones ssh en el servidor, con el siguiente comando en el UFW:

```
# ufw allow 22/tcp
```

Si se quiere acceder a la base de datos PostgreSQL “bloqueada” desde un servidor de páginas web podemos utilizar el siguiente comando para establecer un túnel.

```
$ ssh –L 5511:localhost:5432 usuario@servidordebasedatos.com
```

Una vez hecho esto, el servicio de PostgreSQL será redirigido al puerto número “5511” en el servidor de páginas web, los archivos de conexión en el servidor de páginas web se tendrán que configurar de la siguiente manera (ArPUG, 2014):

Driver.connect("host=localhost port=5111 dbname=sistemabd user=elusuario password=elpassword")
)

La representación gráfica del túnel es la siguiente:

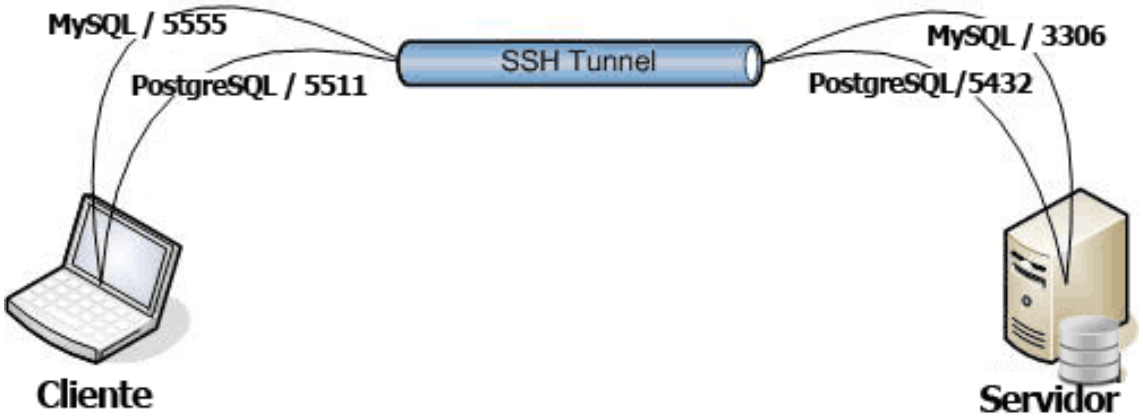

Figura 1 Túnel SSH.

A través del túnel SSH los clientes podrán acceder a los servicios, sin importar que se encuentren bloqueados por sus archivos de configuración. Los sistemas operativos garantizaran la seguridad entre ambos servidores, ya que les solicitará las credenciales pertinentes para el inicio de sesión, SSH garantizará que todas las transmisiones que se hagan entre cliente-servidor sean cifradas.

Este mismo principio se puede aplicar a otros servicios por ejemplo si tuviéramos una base de datos MySQL, se puede generar el siguiente túnel:

$ ssh –L 5555:localhost:3306 usuario@servidordebasedatos.com

Configuración del cliente:

Driver.connect("host=localhost port=5555 dbname=sistemabd user=elusuariomysql password=elpasswordmysql")
)

Para una base de datos Oracle:

$ ssh –L 5555:localhost:1521 usuario@servidordebasedatos.com

Configuración del cliente (Connecting to Oracle Database, 2014):

Driver.connect("host=localhost port=5555 SID=elSID user=elusuariooracle password=elpasswordoracle")

)

Para un servicio de FTP:

$ ssh –L 5566:localhost:21 usuario@servidorftp.com

Configuración del cliente:

Host: localhost

Port: 5566

Usuario: elusuario

Contraseña: lacontraseña

En el caso del servicio FTP que establece una comunicación plana entre el cliente y servidor, al tunelizarlo garantizaremos que ahora los datos se cifren para que ningún usuario malicioso pueda interceptarlos.

Conclusiones

OpenSSH y UFW pueden apoyar a los administradores de servidores en los objetivos de mantener la confidencialidad, integridad y disponibilidad de los datos.

Confidencialidad: se refiere a que se debe proteger la información que sea accedida de alguna parte desconocida, aplicado a cifrar las transmisiones, denegar predeterminadamente los accesos remotos por los archivos de configuración de los servicios.

Integridad. Se refiere a que se debe garantizar la autenticidad de la información, que la información no sea alterada. Los usuarios maliciosos pueden conocer mediante el simple uso del comando nmap, los servicios que se tienen instalados en los servidores. El decir se puede saber si es un servidor de base de datos, de archivos, etc. Conociendo el servicio se sabe sus vulnerabilidades y el método de ataque (DoS, Man in the Middle, Fuerza Bruta, etc). SSH al encriptar las transmisiones hace casi imposible que un atacante descifre la comunicación.

Disponibilidad: Se refiere a que la información debe ser accesible a los usuarios autorizados. En este artículo se propone bloquear todo y únicamente dar acceso a los servicios de SSH y crear túneles para redirigir el tráfico de los servicios bloqueados a través de un canal seguro.

Referencias

[1] Ley Federal de Protección de Datos Particulares en Posesión de los Particulares (5/07/2010). *Cámara de Diputados www.diputados.gob.mx*. http://www.diputados.gob.mx /LeyesBiblio/pdf/LFPDPPP.pdf

[2] PostgreSQL 9.4 Documentation (30/11/2014). *PostgreSQL http://www.postgresql.org* Recuperado de http://www.postgresql.org/docs/9.4/static/auth-pg-hba-conf.html

[3] Ubuntu Official Documentarion (5/12/2014). *PostgreSQL http:// help.ubuntu.com/* Recuperado de https://help.ubuntu.com/community/SSH/OpenSSH/PortForwarding

[4] ArPUD (5/12/2014). *Grupo de usuarios PostgreSQL de Argentina.* http://www.postgresql.org.ar/trac/wiki/InterfasesConexion

[5] Connecting to Oracle Database (2013). Oracle® Data Provider for .NET Developer's Guide Recuperado de http://docs.oracle.com/html/E10927_01/featConnecting.htm