GuacaVPN

Implementación de un acceso centralizado seguro a equipos remotos con Guacamole y OpenVPN

OPENVPN™

Apache Guacamole

Pedro C. aka s4ur0n (@NN2ed_s4ur0n) · CS^3 Group · https://cs3group.com · Versión 1.1

Indice

Agradecimientos

Desde aquí simplemente dar las gracias por seguir trabajando como he estado casi media vida en remoto, siempre buscando las mejores soluciones que sean seguras y nos permitan seguir con la continuidad del negocio. En este caso y motivados por la pandemia mundial del COVID-19, esta guía es simplemente bajo soluciones de código de fuentes abiertas. Ni son las mejores ni son las peores, pero es un avance para que cualquier tipo de organización pueda plantearse poder enviar a toda su fuerza laboral a “casa”. #QuédateEnCasa

No estarán tod@s ni son tod@s las personas a las que me gustaría dedicarles esta guía, pero espero que me comprendáis que es imposible enumerar a tanta gente como conozco.

- Para el gran equipo que forma a **CS3 Group** con los mejores profesionales siempre pendientes de cualquier incidencia de seguridad que pudiera haber y respondiendo ante ellas en tiempo récord.

- A nuestros “viejos sysops de las mejores BBS de España que nos «dejaron dar guerra» cuando necesitábamos aprender de otros” como Belky (https://twitter.com/belky318) y Héctor que no tiene twitter y le pasa como a mí hace algunos años pero lo encontraréis en telegram bajo @Flecman (siempre al pie del cañón y por eso os responderá pasadas 48-76 horas mínimas).

- A toda la gente que está en los hospitales y servicios esenciales que “han caído” y parte que siguen recuperándose con éxito dando parte de sus vidas por salvar otras de esta tremenda “cosa”.

- Por supuesto, a mi familia que me han permitido estar “aislado” escribiendo esta guía que espero que sirva como base para poder implementar todo lo que se trata en ella.

GuacaVPN

1

Acceder a los equipos de trabajo de forma segura, centralizada y remotamente puede no ser una tarea fácil, tener que recordar usuarios, contraseñas, depender de aplicaciones propietarias, exponer puertos en Internet...

Introducción

GuacaVPN

1. Introducción.
2. Objetivos.
3. ¿Qué es Apache Guacamole?
4. Componentes.

1. Introducción

Lamentablemente la situación debido a la pandemia mundial del ***COVID-19 (SARS-CoV-2)*** se encuentra obligando a muchas pequeñas y medianas organizaciones a sufrir situaciones límite, incluso con los temidos ERTE (despidos temporales) ya que muchas no estaban preparadas para poder mantener a toda su fuerza laboral activamente con el "trabajo a distancia" siempre que la actividad lo permita desde cualquier punto del planeta con conexión a Internet.

Esta guía, permitirá instalar un acceso remoto seguro y centralizado para poder continuar con la actividad empresarial basándonos en recursos "open source" (código diseñado de manera que sea accesible al público: todos pueden ver, modificar y distribuir el código de la forma que consideren conveniente) aunque no por ello "sea todo gratis" conforme se asocia.

Con las herramientas empleadas, el coste de la implantación de la solución de acceso remota segura, será el esfuerzo en horas que emplearemos para conseguir nuestro objetivo.

Ha sido diseñada por ***Pedro C. aka s4ur0n*** (@NN2ed_s4ur0n) de ***CS3 Group*** (https://cs3group.com) para devolver a la Comunidad aquello que le permitió a él aprender de otras personas que compartían su trabajo y poder continuar el ciclo.

2. Objetivos

Nuestros objetivos serán:

- Acceder sin ningún cliente específico a todos los equipos de la organización, empleando para ello un navegador compatible con HTML5.
- El acceso será centralizado, mediante autenticación de usuarios, incluso empleado soluciones de 2FA -doble factor de autenticación-, que permita monitorizar los accesos y registrar las conexiones.
- Emplear una VPN (Red Privada Virtual) tanto para las conexiones de los clientes como para el acceso a los equipos remotos con direccionamiento IP privado (RFC-1918) que además, nos permita navegar en Internet.
- No exponer puertos críticos directamente de los equipos de la organización, en principio para RDP -Remote Desktop Protocol- (TCP/3389), SSH Secure SHell- (TCP/22), VNC -Virtual Network Computer- (TCP/5901) y Telnet (TCP/21).
- Emplear soluciones Open Source.
- Emplear protocolos considerados seguros para el acceso a cualquier componente empleado.

3. ¿Qué es Apache Guacamole?

Apache Guacamole es una herramienta de código abierto soportada por la Fundación Apache que se basa en ofrecernos un acceso remoto centralizado bajo unas únicas credenciales para acceder a nuestros equipos de la organización empleando los protocolos más comunes como RDP, VNC, SSH, Telnet e incluso acceso a Kubernetes.

Con ella, simplemente mediante un navegador que soporte HTML5 y Javascript, podremos conectar desde cualquier punto del planeta siempre y cuando contemos lógicamente con un acceso a Internet. Esto lo extiende a poder emplear para el acceso, además de portátiles y equipos de sobremesa, la posibilidad de tabletas y teléfonos.

Su gestión de usuarios centralizada, nos permite resolver los accesos particulares de cada uno y registrar sus operaciones manteniendo un fichero de registro. Incluso contamos con funcionalidades que nos permiten grabar en vídeo las sesiones e incluso las pulsaciones de teclado realizadas. Y no, no hablamos de un troyano “keylogger” sino de poder mantener el control sobre equipos donde es necesario tener el control y registrar en tiempo real lo que se lleva a cabo.

Otras funcionalidades pueden ser la posibilidad de compartir la sesión con otras personas que no disponen de cuenta en el sis-

tema -por ejemplo para situaciones de docencia online- o similares.

Sus características permiten incluso que puedan ofrecerse funcionalidad para copiar y pegar de forma remota, transferir ficheros, emplear el audio remoto, impresión remota y/o local, etc. o bien, puede darse el caso en entornos más restrictivos para evitar filtraciones de información y donde no deseemos que estas funcionalidades sean empleadas y las eliminemos, evitando de esta forma tener que realizar operaciones directamente sobre los equipos remotos.

Además, emplearemos también ***OpenVPN***, otra solución de código abierto ampliamente empleada mundialmente. Este es un sistema cliente/servidor de VPN (Red Privada Virtual) disponible para muchas plataformas que nos permitirá conectar de forma segura -mediante un túnel- a internet o a los equipos remotos desde Microsoft Windows, GNU/Linux, OSX y por supuesto para Android y iOS en el caso de soluciones móviles.

Aprovecharemos también esta solución para poder conectarnos a internet desde nuestra propia red cableada, WiFi o incluso de telefonía móvil (2G/3G/4G ó 5G) para poder hacer uso de internet mediante la VPN donde todo nuestro tráfico se encontrará cifrado a través de un túnel desde nuestro dispositivo en el que conectamos al servidor y desde ahí, saldrá a Internet.

También necesitaremos emplear un servidor ***Apache Tomcat*** que es un servidor web de código abierto desarrollado también por la Fundación Apache para tener nuestro entorno de servidor web HTTP.

Se recomienda encarecidamente la lectura de la documentación del proyecto disponible en la página web oficial en https://guacamole.apache.org/

4. Componentes

Los componentes empleados por ***Apache Guacamole*** no son de una aplicación web monolítica sino que se encuentra compuesto por diferentes partes.

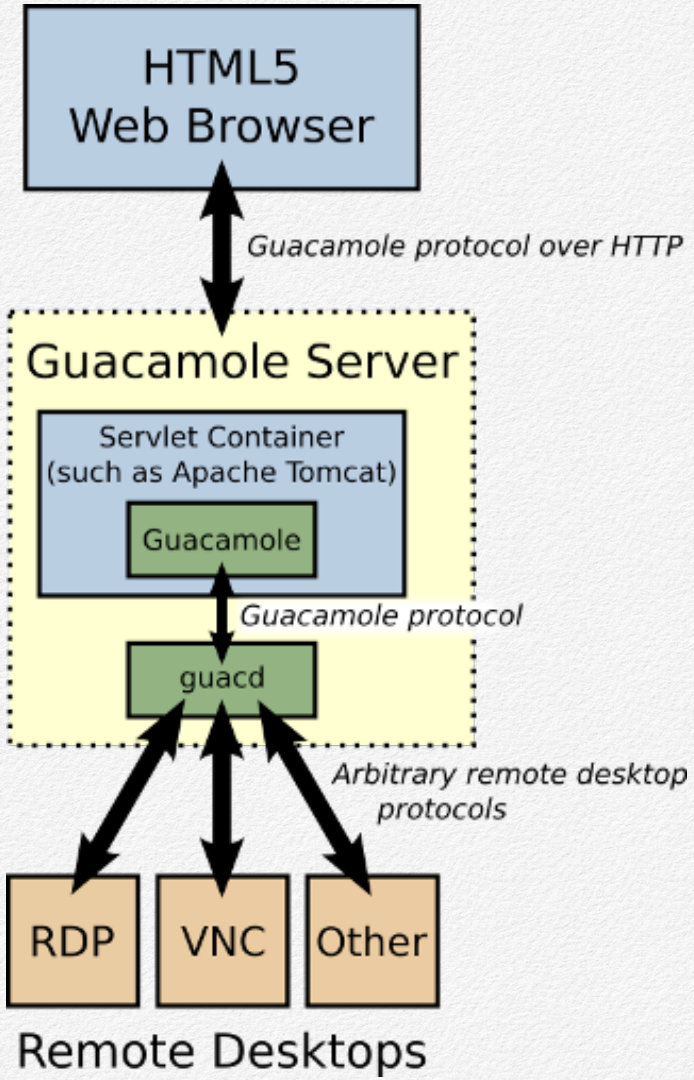
Se trata de una aplicación cliente/servidor. El servidor contiene todos los componentes nativos del lado del servidor requeridos por Guacamole para conectarse a escritorios remotos y el cliente que contiene todos los componentes Java y JavaScript de Guacamole (guacamole, guacamole-common, guacamole-ext y guacamole-common-js). Estos componentes conforman la aplicación web que servirá al cliente HTML5 Guacamole a los usuarios que se conectan a su servidor.

En un entorno de alta disponibilidad, podremos separar dichos componentes en diferentes servidores para balancear las cargas de trabajo (no cubierto por esta guía de implementación).

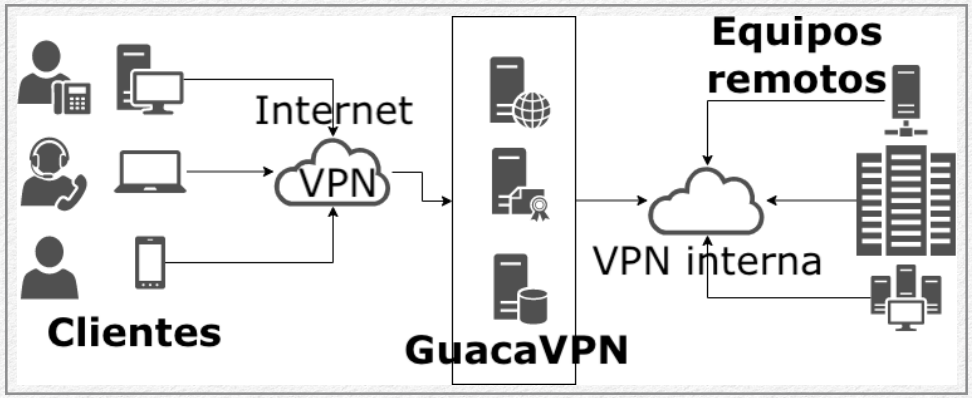
Este tipo de soluciones de alta disponibilidad, al requerir de consultoría especializada para garantizar las cargas, la estabilidad y la disponibilidad, pueden ser ofrecidas por consultores independientes o por ***CS3 Group*** en su caso.

Incluso con los propios ejemplos disponibles en la documentación oficial del proyecto puede montarse y desplegarse mediante Docker de forma muy sencilla.

La implementación y arquitectura de ***Guacamole***, pueden verse en la siguiente figura:



En la arquitectura que emplearemos, se distingue la conexión entre el “cliente remoto” mediante el browser y que será realizada mediante HTTPS (HSTS) -opcionalmente encapsulada en un túnel mediante la VPN- y la parte de conexión a los equipos interna, donde los protocolos RDP, VNC, SSH, Telnet, etc. son transportados mediante la VPN al destino final conforme a la siguiente figura:



Nuestro servidor ***GuacaVPN***, se encontrará alojado en un ISP -Proveedor de Servicios de Internet- para garantizar su máxima disponibilidad y que pueda ser alcanzado desde cualquier punto de internet. Dispondrá de 2 interfaces de red -una externa para la conexión de los clientes- y otra interna para conexión exclusiva de la VPN que crearemos con los equipos remotos a conectar.

La elección del proveedor, la dejamos en manos del lector, puesto que hoy en día, proveedores como OVH, Digital Ocean, etc. ofrecen este tipo de soluciones como VPS -Servidor Privado Virtual- a un coste muy reducido.

Las características de dicho servidor vendrán determinadas en función de la carga que deba soportar, teniendo principalmente en cuenta la conectividad externa e interna más que en la memoria o capacidad de almacenamiento contratado.

Incluso este mismo servidor, puede ser interno y hacer una redirección a los puertos que empleemos para atender a los clientes.

Para la guía, hemos elegido un servidor basado en la última versión de Debian 10 “buster” de 64 bits partiendo de la instalación mínima “netinstall” y habilitando su acceso por SSH con las utilidades estándar de sistema del menú de instalación.

No es objetivo de esta guía instalar el sistema operativo, por lo que si no se hubiera hecho nunca, puede consultarse una completa guía para su instalación en https://www.howtoforge.com/tutorial/debian-minimal-server/

GuacaVPN

2

Compilar y no morir en el intento...

Compilación

GuacaVPN

1. **Nuestras imprescindibles.**
2. **Herramientas comunes.**
3. **Compilación del servidor.**
4. **Compilación del cliente.**
5. **Bash script.**

1. Nuestras imprescindibles

Antes de proceder a la compilación desde el propio código fuente del servidor y del cliente, instalaremos nuestras herramientas favoritas en el sistema.

Por ejemplo, podremos añadir:

```
apt -y install vim net-tools bmon htop tcpdump
traceroute hping3
```

2. Herramientas comunes

Tenemos que instalar los paquetes de Debian 10 imprescindibles para poder compilar con éxito el servidor y el cliente.

Se ha pretendido poder compilar con todos los protocolos disponibles -en su momento RDP, VNC, SSH, Telnet y soporte de Kubernetes- pero algunas de las siguientes librerías no serían necesarias si no se necesitase.

Procederemos con ello:

```
apt -y install maven default-jdk git libcairo2-dev libjpeg62-turbo-dev libwebsockets-dev libpng-dev libossp-uuid-dev gcc make dh-autoreconf tomcat9 tomcat9-admin tomcat9-user libavcodec-dev libavutil-dev libswscale-dev freerdp2-dev libpango1.0-dev libssh2-1-dev libtelnet-dev libvncserver-dev libpulse-dev libssl-dev libvorbis-dev libwebp-dev
```

3. Compilar el servidor

Procederemos a descargar la versión desde github ya que se encontrará siempre actualizado, aunque otra forma, podría ser descargar directamente el código fuente desde su página del proyecto oficial:

```
cd /root

wget https://downloads.apache.org/guacamole/1.1.0/source/guacamole-server-1.1.0.tar.gz

tar -xzf guacamole-server-1.1.0.tar.gz

cd guacamole-server-1.1.0/
```

Sin embargo, lo haremos desde su repositorio de github:

```
cd /root

git clone git://github.com/apache/guacamole-server.git

cd guacamole-server/
```

Ahora, debido a que la versión de github no tiene disponible el fichero “configure”, vamos a generarlo:

```
autoreconf -fi
```

Ahora, la configuración será común tanto en uno como en otro. Procederemos pues a compilar el servidor mediante:

```
./configure --with-init-dir=/etc/init.d
make
make install
ldconfig
```

Con ello, ya tendremos el servidor preparado para su ejecución.

4. Compilación del cliente

De la misma forma que el servidor, podemos descargar el paquete con el código fuente desde su página oficial del proyecto, aunque nuestra recomendación es hacerlo siempre desde github.

Para ello, tendríamos que escribir:

```
cd /root
git clone
git://github.com/apache/guacamole-client.git
cd guacamole-client
tar -xzf guacamole-client-1.1.0.tar.gz
cd guacamole-client-1.1.0/
```

Para clonar el proyecto desde github, pondremos:

cd /root

git clone git://github.com/apache/guacamole-client.git

cd guacamole-client

Llegados a este punto, será común tanto si hemos descargado el código fuente o hemos clonado el repositorio de github.

Si la instalación como es nuestro caso ha sido desde cero, no tendremos la variable JAVA_HOME establecida, por lo que tendremos que hacerlo mediante:

```
echo
JAVA_HOME="/usr/lib/jvm/java-11-openjdk-amd64/
" >> /etc/environment

source /etc/environment
```

El código fue preparado para compilar con la versión 8 del SDK de Java por lo que necesitará que modifiquemos un par de ficheros ya que la versión actual de Java SDK es la 11 para que no falle.

Buscaremos los ficheros que necesitan ser actualizados con:

```
grep -nrw . -e "maven-javadoc-plugin"
```

Estos serán:

```
./guacamole-common/pom.xml

./guacamole-ext/pom.xml
```

Editaremos los ficheros y añadiremos `<source>8</source>` en la configuración del plugin de ***maven-javadoc-plugin***.

Quedarán conforme se visualiza:

```
<!-- Attach JavaDoc jar -->
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-javadoc-plugin</artifactId>
    <version>2.10.3</version>
    <configuration>
         <detectOfflineLinks>false</detectOfflineLinks>
         <source>8</source>
    </configuration>
    <executions>
        <execution>
            <id>attach-javadocs</id>
            <goals>
                <goal>jar</goal>
            </goals>
        </execution>
    </executions>
</plugin>
```

Ahora ya podemos proceder a compilar el cliente:

```
mvn package

mkdir -p /var/lib/tomcat9/webapps
```

```
cd guacamole/target/

mv guacamole-1.2.0.war guacamole.war

cp guacamole.war /var/lib/tomcat9/webapps/
```

Ahora, reiniciaremos los servicios con:

```
service tomcat9 restart

/etc/init.d/guacd start
```

Crearemos la configuración de guacamole. Para ello, creamos los directorios necesarios mediante:

```
mkdir /etc/guacamole
/usr/share/tomcat9/.guacamole

cd /etc/guacamole
```

Ahora creamos el fichero `guacamole.properties` con el siguiente contenido:

```
guacd-hostname: localhost

guacd-port: 4822

user-mapping: /etc/guacamole/user-mapping.xml

auth-provider:
net.sourceforge.guacamole.net.basic.BasicFileA
uthenticationProvider

basic-user-mapping:
/etc/guacamole/user-mapping.xml
```

Y lo enlazamos en el Tomcat mediante:

```
ln -s guacamole.properties
/usr/share/tomcat9/.guacamole/
```

Generamos ahora un fichero `user-mapping.xml` que contendrá nuestros usuarios básicos y conexiones a los equipos remotos. Todavía no hemos configurado el OpenVPN por lo que pondremos las direcciones IP que podamos alcanzar para realizar la primera prueba.

Su contenido inicial será el siguiente:

```
<authorize
username="cs3group"
password="str0ngP4$$w0rd">
    <connection name="SSH">
        <protocol>ssh</protocol>
        <param name="hostname">1.2.3.4</param>
        <param name="port">22</param>
        <param name="username">s4ur0n</param>
    </connection>
</authorize>
```

Como se observa, el fichero tiene hardcodeadas las credenciales. Posteriormente cuando empleemos el sistema de base de datos, estas quedarán guardadas en ella.

Damos los permisos y el usuario/grupo necesarios al fichero creado mediante:

```
chmod 600 user-mapping.xml
chown tomcat:tomcat user-mapping.xml
```

Reiniciaremos los servicios:

```
service tomcat9 restart
/etc/init.d/guacd restart
```

Y ya podremos acceder vía web a Guacamole -observar que todavía no ha sido instalado ningún certificado “legítimo” y nos conectaremos por http- aunque para las primeras pruebas nos servirá.

Conectaremos a:

`http://ip_servidor:8080/guacamole`

Introduciremos las credenciales indicadas en el fichero `user-mapping.xml` y veremos el acceso por SSH a la máquina de destino.

Si todas las pruebas han sido correctas, podremos continuar con la instalación del resto de los componentes.

5. Bash Script

Puede realizarse todo mediante un pequeño script:

```
#!/bin/bash
#
# Guacamole v1.1.0 (Debian Burst)
# CS3 Group by Pedro C. aka s4ur0n
# @NN2ed_s4ur0n
# info@cs3group.com
#

#
# Favorite tools
#
clear
echo "Installing Favorite Tools..."
apt -y install vim net-tools bmon htop tcpdump traceroute hping3


#
# Guacamole Main Packages
#
clear
echo "Installing Guacamole Main Packages..."
apt -y install maven default-jdk git libcairo2-dev libjpeg62-turbo-
dev libwebsockets-dev libpng-dev libossp-uuid-dev gcc make dh-autore-
conf tomcat9 tomcat9-admin tomcat9-user libavcodec-dev libavutil-dev
libswscale-dev freerdp2-dev libpango1.0-dev libssh2-1-dev libtelnet-
dev libvncserver-dev libpulse-dev libssl-dev libvorbis-dev libwebp-
dev

#
# Guacamole Server
#
clear
echo "Downloading latest Guacamole Server..."
cd /root
#
# From source package:
#
# wget
https://downloads.apache.org/guacamole/1.1.0/source/guacamole-server
-1.1.0.tar.gz
# tar -xzf guacamole-server-1.1.0.tar.gz
# cd guacamole-server-1.1.0/
```

```
#
git clone git://github.com/apache/guacamole-server.git
cd guacamole-server/
# Only for git version: (see
https://guacamole.apache.org/doc/gug/installing-guacamole.html#building-guacamole-client)
autoreconf -fi
./configure --with-init-dir=/etc/init.d
make
make install
ldconfig

#
# Guacamole Client
#
clear
echo "Downloading latest Guacamole Client..."
cd /root
#
#
# wget
https://downloads.apache.org/guacamole/1.1.0/source/guacamole-client-1.1.0.tar.gz
# tar -xzf guacamole-client-1.1.0.tar.gz
# cd guacamole-client-1.1.0/
#
git clone git://github.com/apache/guacamole-client.git
cd guacamole-client
echo JAVA_HOME="/usr/lib/jvm/java-11-openjdk-amd64/" > /etc/environ-
ment
source /etc/environment
clear
echo "You need to patch this files:"
grep -nrw . -e "maven-javadoc-plugin"
echo "Add <source>8</source> below <configuration> at the maven-java-
doc-plugin"
echo " "
echo "Example:"
echo "   <artifactId>maven-javadoc-plugin</artifactId>"
echo "      <version>2.10.3</version>"
echo "                <configuration>"
echo "                  <source>8</source>"
echo "              ..."
read -n 1 -s -r -p "Press any key to continue..."
clear
echo "Creating war package"
mvn package
mkdir -p /var/lib/tomcat9/webapps
cd guacamole/target/
mv guacamole-1.2.0.war guacamole.war
cp guacamole.war /var/lib/tomcat9/webapps/

#
# Restart services
#
clear
echo "Restarting services..."
service tomcat9 restart
/etc/init.d/guacd start

#
# Create config
#
clear
echo "Creating Guacamole config..."
mkdir /etc/guacamole /usr/share/tomcat9/.guacamole
cd /etc/guacamole
echo "
guacd-hostname: localhost
guacd-port: 4822
user-mapping: /etc/guacamole/user-mapping.xml
auth-provider:
net.sourceforge.guacamole.net.basic.BasicFileAuthenticationProvider
basic-user-mapping: /etc/guacamole/user-mapping.xml
" > guacamole.properties
ln -s guacamole.properties /usr/share/tomcat9/.guacamole/

#
# Create service user
#
clear
read -p "Define a user for administration panel: " guacauser
read -s -p "Define a user password: " guacapassword
read -p "Define an IP for the remote host:" ip
read -p "Define a remote user:" ruser
echo '<user-mapping>
<authorize
username="'$guacauser'"
password="'$guacapassword'">
<connection name="SSH">
<protocol>ssh</protocol>
<param name="hostname">'$ip'</param>
<param name="port">22</param>
<param name="username">'$ruser'</param>
</connection>
</authorize>
</user-mapping>' > user-mapping.xml
chmod 600 user-mapping.xml
chown tomcat:tomcat user-mapping.xml
echo 'Remember add in authorize encoding="md5" and generate pass
with printf "%s" "<password>" | md5sum'

#
# Restart main services
#
clear
echo "Restarting main services"
# Restart all main services
service tomcat9 restart
/etc/init.d/guacd restart
```

```
echo "Connect to http://$ip:8080/guacamole with $guacauser and your
password"
echo "See https://guacamole.apache.org/doc/gug/"
echo "See
https://guacamole.apache.org/doc/gug/configuring-guacamole.html for
user-mapping"
```

GuacaVPN

3

Líbranos de compilar y tener errores que no sabemos resolver o nos llevan un día de trabajo...

Sin compilar

Sin Compilar

1. Instalación mínima

1. Instalación mínima

Intentaremos realizar una instalación de Guacamole con su mínima instalación. En este punto, mi particular recomendación es siempre compilarlo y usar todo desde las propias fuentes, pero he encontrado muchos casos donde se necesita realizar una instalación con los mínimos componentes esenciales para que pueda funcionar.

De nuevo, nos basaremos en una instalación basada en Debian 10 “Buster” de 64 bits mediante netinstall (los mínimos componentes) y con las opciones de Servidor SSH y Utilidades estándar del sistema.

Antes de nada, instalaremos nuestras herramientas favoritas:

```
apt -y install vim net-tools bmon htop tcpdump
traceroute hping3
```

Instalaremos las librerías y paquetes mínimos para poder montar nuestro entorno de trabajo. En este caso, emplearemos como componentes añadidos la base de datos ***MySQL Server*** ya

que en este caso, gestionaremos a varios grupos de usuarios para el acceso remoto.

Escribiremos los siguientes comandos:

```
apt update
apt -y upgrade
apt -y install git tomcat9 tomcat9-admin
tomcat9-user build-essential default-mysql-ser-
ver default-mysql-client mysql-common
freerdp2-dev libcairo2-dev libjpeg62-turbo-dev
libpng-dev libossp-uuid-dev libavcodec-dev li-
bavutil-dev freerdp2-dev libpango1.0-dev
libssh2-1-dev libtelnet-dev libvorbis-dev
libwebp-dev libswscale-dev libvncserver-dev
libpulse-dev libssl-dev libwebsockets-dev
```

Es importante recordar que en ésta guía, no nos preocuparemos de la fortificación de la base de datos, lo que dejamos en manos del lector para hacer, pero al menos, consideraremos el siguiente script para eliminar algunos puntos vulnerables:

```
mysql_secure_installation
```

Y cambiaremos la contraseña del usuario root y el acceso remoto ya que no será necesario en nuestro caso.

A continuación, descargaremos algunos componentes de Guacamole y aunque se trate de la instalación mínima, al menos el componente del servidor si que será necesario compilarlo aunque lo bueno, es que nos ahorraremos la instalación del SDK de Java, el gestor maven, ant, etc...

Estos paquetes, en el momento de escribir la guía, son el último fichero war de guacamole (el cliente), el servidor que lo haremos desde su propio repositorio de github (aunque puede emplearse el paquete con el código fuente disponible en la página oficial del proyecto), el paquete ***guacamole-auth-jdbc*** y el ***connector jdbc*** de mysql (aunque podríamos emplear el paquete deb disponible desde la página del proyecto de MySQL en https://dev.mysql.com/downloads/connector/j/ e instalarlo mediante dpkg. Estos dos últimos, son imprescindibles para el acceso a la base de datos.

No olvidemos que en el caso de descargar de fuentes externas, debemos posteriormente comprobar mediante OpenPGP las sumas de comprobación de los ficheros. Revisar dichas firmas en https://guacamole.apache.org/releases/1.1.0/

Procederemos con ello:

```
cd /tmp
wget
http://apache.mirrors.ionfish.org/guacamole/1.
1.0/binary/guacamole-1.1.0.war
git clone
git://github.com/apache/guacamole-server.git
```

```
wget
http://apache.mirrors.ionfish.org/guacamole/1.
1.0/binary/guacamole-auth-jdbc-1.1.0.tar.gz
wget
http://ftp.iij.ad.jp/pub/db/mysql/Downloads/Co
nnector-J/mysql-connector-java-8.0.19.tar.gz
mkdir -p /etc/guacamole/lib
mkdir -p /etc/guacamole/extensions
cd /tmp/guacamole-server
autoreconf -fi
./configure --with-init-dir=/etc/init.d
make
make install
ldconfig
systemctl enable guacd
cd /tmp
mv guacamole-1.1.0.war
/etc/guacamole/guacamole.war
ln -s /etc/guacamole/guacamole.war /var/lib/
tomcat9/webapps/
cd /tmp
tar -zxvf mysql-connector-java-8.0.19.tar.gz
tar -zxvf guacamole-auth-jdbc-1.1.0.tar.gz
cd mysql-connector-java-8.0.19/
mv mysql-connector-java-8.0.19.jar /etc/guaca-
mole/lib/
cd /tmp/guacamole-auth-jdbc-1.1.0/mysql
mv guacamole-auth-jdbc-mysql-1.1.0.jar /etc/
guacamole/extensions/
```

Con esto, tendremos ya todo preparado. Ahora nos tocará configurar Guacamole y Apache Tomcat. Escribiremos:

```
echo "GUACAMOLE_HOME=/etc/guacamole" >> /etc/
default/tomcat9
```

Ahora, configuraremos el acceso a la Base de Datos:

```
mysql -u root -p

MariaDB [(none)]> create database guacamo-
le_db;

MariaDB [(none)]> create user 'guacauser'@'lo-
calhost' identified by "p4$$w0rd";

MariaDB [(none)]> GRANT SELECT,INSERT,UPDA-
TE,DELETE ON guacamole_db.* TO 'guacauser'@'lo-
calhost';

MariaDB [(none)]> flush privileges;

MariaDB [(none)]> exit;
```

Añadiremos los esquemas correspondientes:

```
cd /tmp/guacamole-auth-jdbc-1.1.0/mysql/schema

cat *.sql | mysql -u root -p guacamole_db
```

Configuraremos Guacamole para que tenga acceso a la Base de Datos mediante el fichero `guacamole.properties`

```
cd /etc/guacamole/

vim guacamole.properties

...

# Disable other auth-provider with a comment

auth-provider:
net.sourceforge.guacamole.net.auth.mysql.MySQL
AuthenticationProvider

# MySQL options

mysql-hostname: localhost

mysql-port: 3306

mysql-database: guacamole_db

mysql-username: guacauser

mysql-password: p4$$w0rd

# Disable duplicate connections

mysql-disallow-duplicate-connections: true

...
```

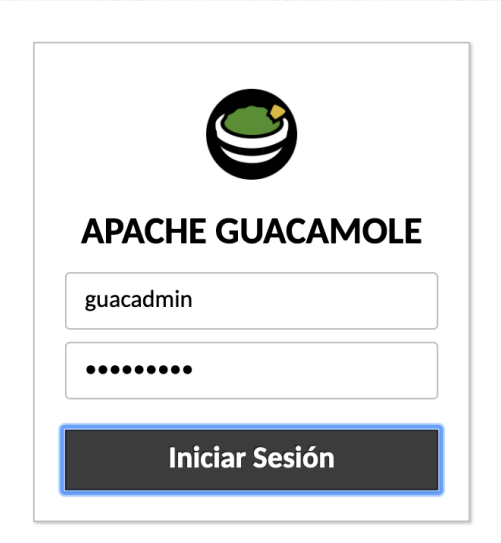
Y finalmente, lo enlazaremos y reiniciaremos los servicios de Tomcat y Guacamole Server:

```
ln -s /etc/guacamole
/usr/share/tomcat9/.guacamole

service tomcat9 restart

service guacd restart

service tomcat9 status

service guacd status
```

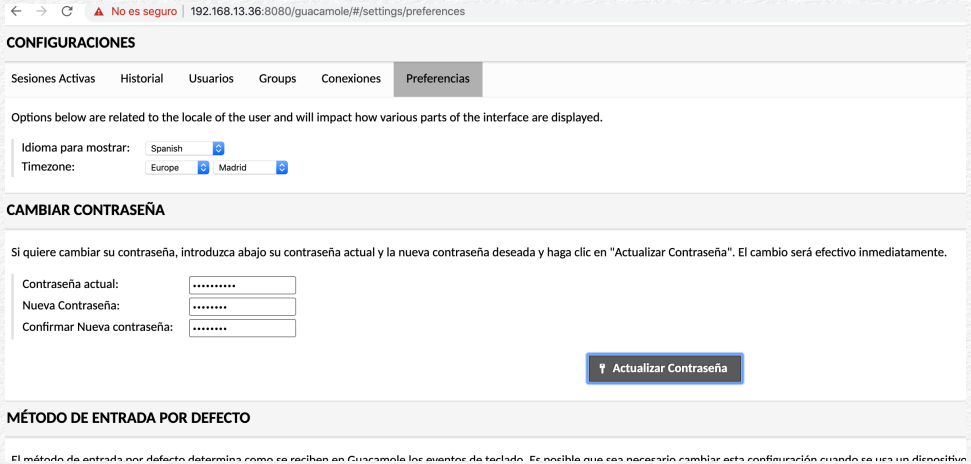
Desde un navegador, podremos conectar ya a la dirección IP por http en el puerto 8080 y la ruta /guacamole.

```
http://a.b.c.d:8080/guacamole/
```

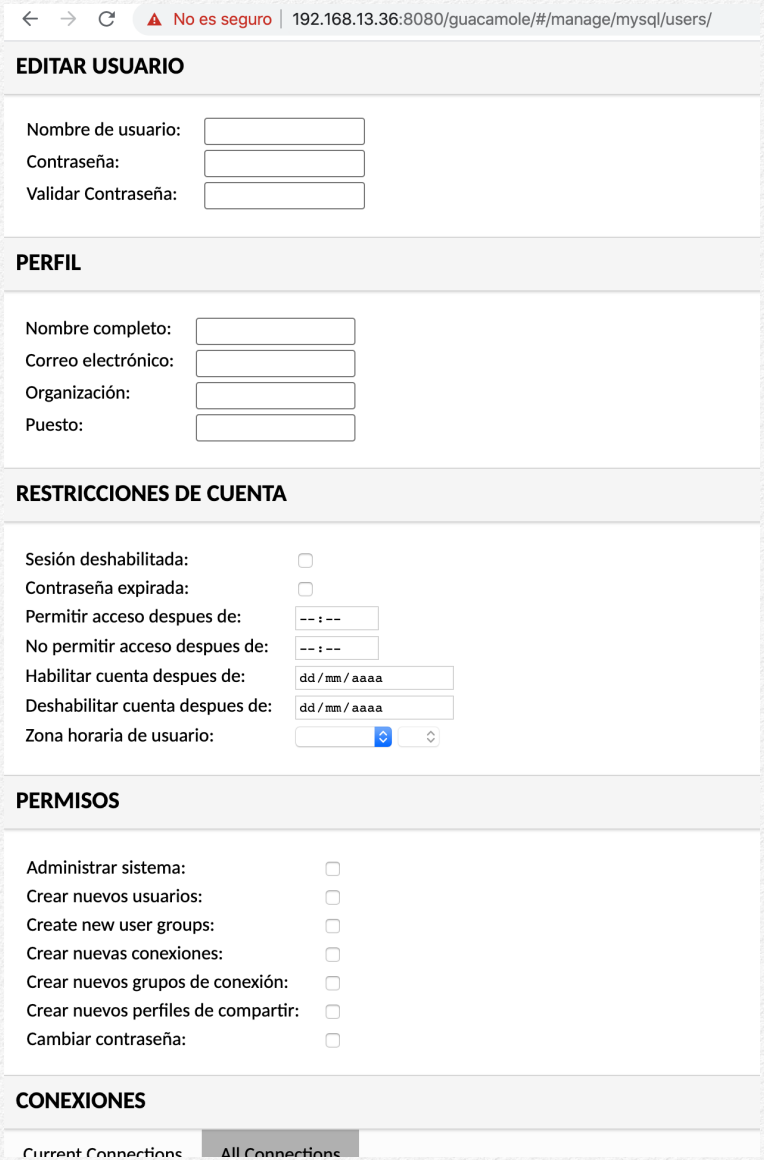
Podremos entrar con el usuario **guacadmin** con la contraseña **guacadmin**



Una vez hecho, iremos a la esquina superior derecha y en nuestro usuario, nos iremos a configuración, preferencias y cambiaremos la contraseña.



Desde este panel, ya podremos cambiar las preferencias, crear y gestionar los grupos, usuarios, etc...



Es posible que al intentar cambiarlo, nos de un error de permisos y no nos deje.

Si ese fuera nuestro caso, procederemos directamente a actualizarlo en la base de datos. Procederemos como se explica:

```
mysql -u root -p

MariaDB [(none)]> use guacamole_db;


MariaDB [guacamole_db]> SET @salt =
UNHEX(SHA2(UUID(), 256));

Query OK, 0 rows affected (0.000 sec)


MariaDB [guacamole_db]> UPDATE guacamole_user
SET entity_id=(SELECT entity_id FROM guacamo-
le_entity WHERE name = 'guacadmin' AND type =
'USER'), password_salt=@salt,
password_hash=UNHEX(SHA2(CONCAT('p4$$w0rd',
HEX(@salt)), 256)), password_date=CURRENT_TIME-
STAMP;

Query OK, 1 row affected (0.004 sec)

Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [guacamole_db]> exit;
```

Desconectaremos y comprobaremos el cambio con la nueva contraseña asignada para poder seguir trabajando.

Toda la documentación al respecto de MySQL puede encontrarse en la página oficial https://guacamole.apache.org/doc/gug/jdbc-auth.html y para aquellos lectores que deseen realizarlo bajo otras bases de datos como PostgreSQL, SQL Server, etc... podrán seguir las guías oficiales disponibles.

También es importante configurarlo con las opciones de seguridad que permite como la complejidad de las credenciales, caducidad, prevención de volver a emplear la misma contraseña, conexiones concurrentes, etc.

Dejamos al lector para que pueda definir una política de seguridad en su acceso de acuerdo a las necesidades de su organización, no haciendo esta guía de dichas características aunque en la práctica, si que son empleadas en los despliegues realizados.

GuacaVPN

4

Compilación y autenticación centralizada con Base de Datos (MySQL/MariaDB)

Compilado y con autenticación en BB.DD.

Pasos

1. Instalación completa

1. Instalación completa

Partiendo de nuestra compilación (ver capítulo 1) y la instalación en este caso de ***MySql o MaríaDB*** sin entrar en detalles de su securización completa, básicamente lo que haremos es crear la infraestructura necesaria para que Guacamole nos permita una autenticación para los usuarios basada en dicha Base de Datos.

No obstante, con la documentación oficial del proyecto disponible en https://guacamole.apache.org/doc/gug/jdbc-auth.html podemos realizarlo sobre otras como PostgreSQL o un SQL Server.

Partiremos de una instalación básica de MySQL:

```
apt -y install default-mysql-server default-
mysql-client mysql-common
```

Es importante recordar que en ésta guía, no nos preocuparemos de la fortificación de la base de datos, lo que dejamos en manos del lector para hacer, pero al menos, consideraremos el siguiente script para eliminar algunos puntos vulnerables:

```
mysql_secure_installation
```

Y cambiaremos la contraseña del usuario root y el acceso remoto ya que no será necesario en nuestro caso.

Ahora, necesitaremos instalar el paquete ***guacamole-auth-jdbc*** y el ***conector jdbc*** de mysql. En esta instalación, lo que haremos será eliminar completamente la que hubieramos tenido anteriormente para las pruebas, partiendo prácticamente de cero para regenerar todo a partir de nuestros ficheros compilados (respetaremos el /etc/default/tomcat9 por si hubieran instaladas otras aplicaciones). Por lo tanto, pondremos:

```
service tomcat9 stop
service guacd stop
rm -fr /etc/guacamole/
rm -fr /usr/share/tomcat9/.guacamole
rm -fr /var/lib/tomcat9/webapps/guacamole*
cd /root/guacamole-server
autoreconf -fi
./configure --with-init-dir=/etc/init.d
make
make install
ldconfig
systemctl enable guacd
cd /root/guacamole-client
mvn package
mkdir -p /etc/guacamole
mkdir -p /etc/guacamole/lib
mkdir -p /etc/guacamole/extensions
mkdir -p /var/lib/tomcat9/webapps
mkdir -p /usr/share/tomcat9/.guacamole
cp guacamole/target/guacamole-1.2.0.war
/etc/guacamole/
mv /etc/guacamole/guacamole-1.2.0.war
/etc/guacamole/guacamole.war
ln -s /etc/guacamole/guacamole.war /var/lib/
tomcat9/webapps
cd
/root/guacamole-client/extensions/guacamole-au
th-jdbc/modules/guacamole-auth-jdbc-mysql
```

```
cp target/guacamole-auth-jdbc-mysql-1.2.0.jar
/etc/guacamole/extensions/
```

Ahora, configuraremos el acceso a la Base de Datos:

```
mysql -u root -p

MariaDB [(none)]> create database guacamo-
le_db;

MariaDB [(none)]> create user 'guacauser'@'lo-
calhost' identified by "p4$$w0rd";

MariaDB [(none)]> GRANT SELECT,INSERT,UPDA-
TE,DELETE ON guacamole_db.* TO 'guacauser'@'lo-
calhost';

MariaDB [(none)]> flush privileges;

MariaDB [(none)]> exit;
```

Añadiremos los esquemas correspondientes:

```
cd schema

cat *.sql | mysql -u root -p guacamole_db
```

Ahora, descargaremos el conector para MySQL:

```
cd /root

wget
http://ftp.iij.ad.jp/pub/db/mysql/Downloads/Co
nnector-J/mysql-connector-java-8.0.19.tar.gz

tar -zxvf mysql-connector-java-8.0.19.tar.gz

rm mysql-connector-java-8.0.19.tar.gz

cd mysql-connector-java-8.0.19

cp mysql-connector-java-8.0.19.jar /etc/guaca-
mole/lib/
```

Configuraremos Guacamole para que tenga acceso a la Base de Datos mediante el fichero `guacamole.properties`

```
cd /etc/guacamole/

vim guacamole.properties

auth-provider:
net.sourceforge.guacamole.net.auth.mysql.MySQL
AuthenticationProvider
```

```
# MySQL options

mysql-hostname: localhost

mysql-port: 3306

mysql-database: guacamole_db

mysql-username: guacauser

mysql-password: p4$$w0rd

# Disable duplicate connections

mysql-disallow-duplicate-connections: true
```

Emplearemos también el mismo uso horario los servidores empleados en Guacamole y MySQL si se encontrasen en distintos contenedores o servidores, ya que deberán de encontrarse coordinados (preferiblemente para emplear UTC) o bien, mediante nuestra “hora local” preferida. Especialmente si empleamos nuestro driver compilado (la versión 1.2.0 en estos momentos) ya que en otro caso, es muy posible obtener un error de autenticación debido a ello empleando el CEST (horario de verano) en los sitios donde se emplea. Este error será similar a:

```
cat /var/log/tomcat9/catalina.out

...

[info] ### Cause: java.sql.SQLException: The
server time zone value 'CEST' is unrecognized
or represents more than one time zone. You
must configure either the server or JDBC dri-
ver (via the 'serverTimezone' configuration
property) to use a more specifc time zone va-
lue if you want to utilize time zone support.
```

En ese caso, en el servidor de bases de datos, cargaremos las zonas horarias mediante:

```
mysql_tzinfo_to_sql /usr/share/zoneinfo |
mysql -u root mysql
```

Y fijaremos su empleo de forma global:

```
mysql -u root -p

MariaDB [mysql]> SET
@@global.time_zone='Europe/Madrid';

Query OK, 0 rows affected (0.001 sec)

MariaDB [mysql]>exit;
```

Además, en el sistema pondremos:

```
timedatectl set-timezone Europe/Madrid
```

Añadiremos dicha configuración de forma global para MySQL/MariaDB en nuestro sistema mediante:

```
echo -e "default_time_zone='Europe/Madrid'" >>
/etc/mysql/mariadb.conf.d/50-server.cnf

service mysql restart
```

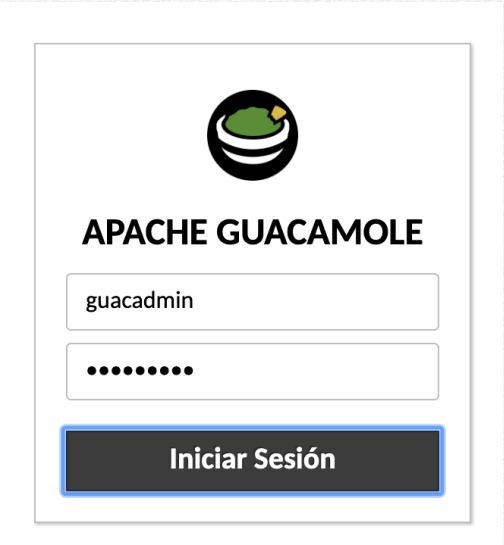
Finalmente, lo enlazaremos e iniciaremos los servicios de Tomcat y Guacamole Server:

```
ln -s /etc/guacamole
/usr/share/tomcat9/.guacamole

service tomcat9 start

service guacd start

service tomcat9 status

service guacd status
```

Desde un navegador, podremos conectar ya a la dirección IP por http en el puerto 8080 y la ruta /guacamole.

```
http://a.b.c.d:8080/guacamole/
```

Podremos entrar con el usuario **guacadmin** con la contraseña **guacadmin**



Si hemos seguido todos los pasos, veremos que podemos acceder pero no podremos cambiar la contraseña debido a un error de permisos.

Aprovecharemos para introducir una nueva contraseña más robusta que posteriormente cumpla con las políticas de nuestra organización:

```
mysql -u root -p

MariaDB [(none)]> use guacamole_db;


MariaDB [guacamole_db]> SET @salt =
UNHEX(SHA2(UUID(), 256));

Query OK, 0 rows affected (0.000 sec)


MariaDB [guacamole_db]> UPDATE guacamole_user
SET entity_id=(SELECT entity_id FROM guacamo-
le_entity WHERE name = 'guacadmin' AND type =
'USER'), password_salt=@salt,
password_hash=UNHEX(SHA2(CONCAT('P4$$@w0Rd-',
HEX(@salt)), 256)), password_date=CURRENT_TIME-
STAMP;

Query OK, 1 row affected (0.004 sec)

Rows matched: 1  Changed: 1  Warnings: 0


MariaDB [guacamole_db]> exit;
```

Desconectaremos y comprobaremos el cambio con la nueva contraseña asignada para poder seguir trabajando.

Volveremos a modificar el fichero `/etc/guacamole/guacamole.properties` para añadir algo más de complejidad. Será algo parecido a:

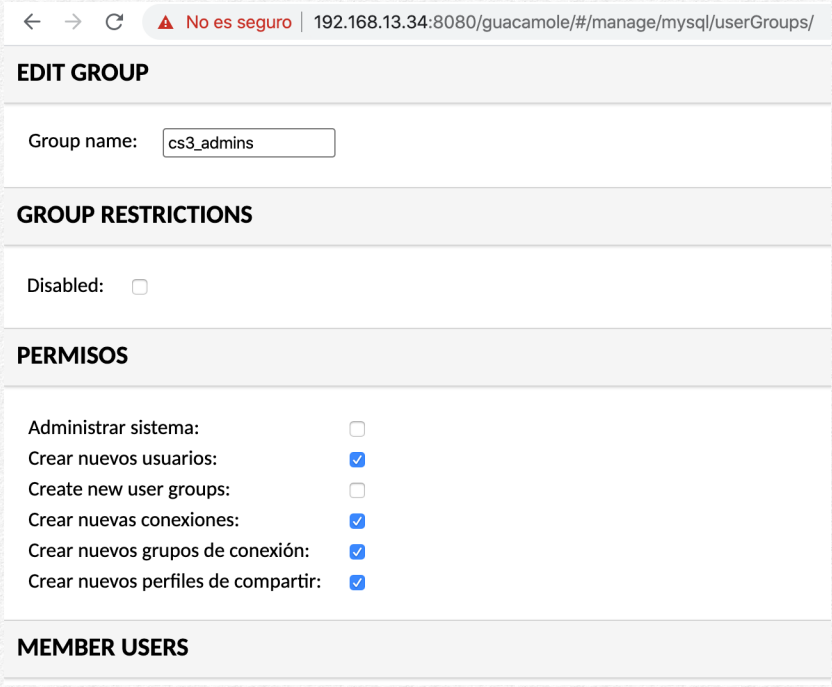
```
auth-provider:
net.sourceforge.guacamole.net.auth.mysql.MySQL
AuthenticationProvider

# MySQL options
mysql-hostname: localhost
mysql-port: 3306
mysql-database: guacamole_db
mysql-username: guacauser
mysql-password: p4$$w0rd

# Password complexity
mysql-user-password-min-length: 8
mysql-user-password-require-multiple-case:
true
mysql-user-password-require-symbol: true
mysql-user-password-require-digit: true
```

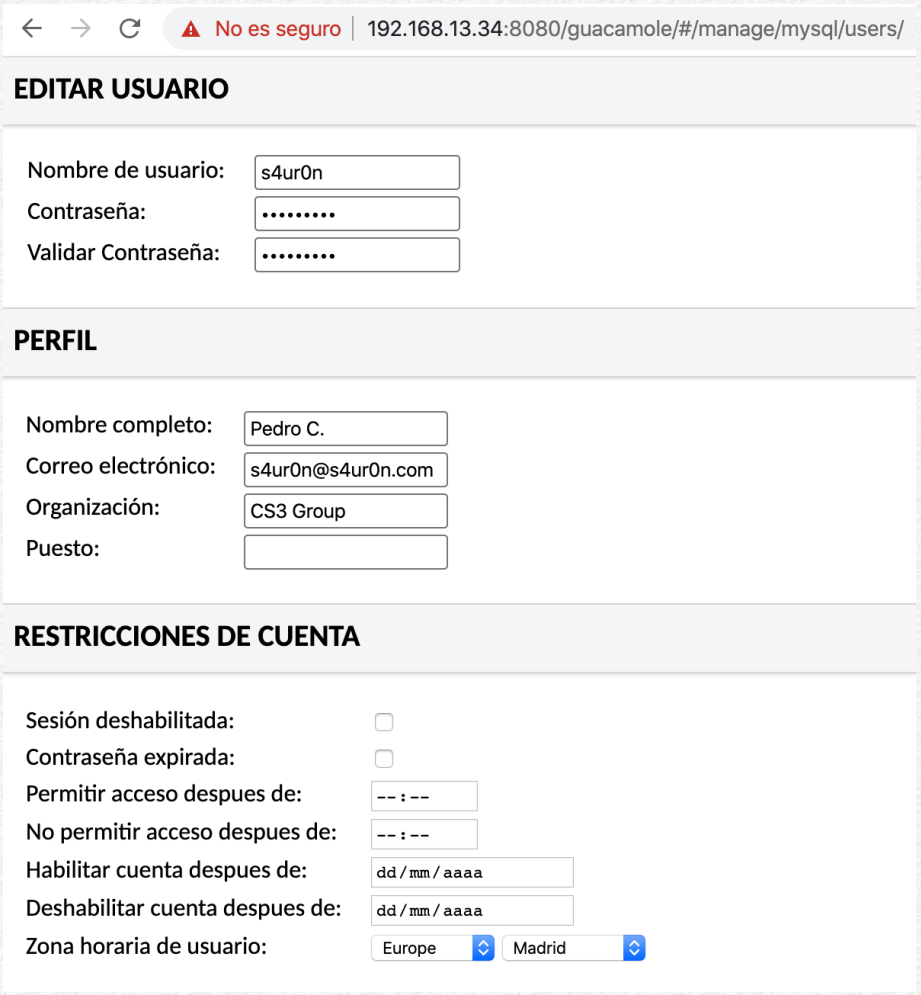
```
mysql-user-password-prohibit-username: true

# Password age/expiration
mysql-user-password-min-age: 7
mysql-user-password-max-age: 30

# Preventing password reuse
mysql-user-password-history-size: 12

# Concurrent use of Guacamole connections
mysql-default-max-connections: 1
mysql-default-max-group-connections: 1

# Disable duplicate connections
mysql-disallow-duplicate-connections: true

# Restricting authentication to database users
only
mysql-user-required: true
```

Reiniciaremos Tomcat9 y se aplicarán los nuevos criterios:

```
service tomcat9 restart
```

Ya podremos entonces gestionar los grupos, usuarios, conexiones, etc. desde su interface gráfica conforme a las figuras:




No es seguro | 192.168.13.34:8080/guacamole/#/manage/mysql/users/
EDITAR USUARIO
Nombre de usuario: s4ur0n
Contraseña: ••••••••
Validar Contraseña: ••••••••
PERFIL
Nombre completo: Pedro C.
Correo electrónico: s4ur0n@s4ur0n.com
Organización: CS3 Group
Puesto:
RESTRICCIONES DE CUENTA
Sesión deshabilitada:
Contraseña expirada:
Permitir acceso despues de: --:--
No permitir acceso despues de: --:--
Habilitar cuenta despues de: dd/mm/aaaa
Deshabilitar cuenta despues de: dd/mm/aaaa
Zona horaria de usuario: Europe Madrid

GuacaVPN

5

Dando unos ligeros retoques estéticos y de seguridad a nuestra instalación

Retoques estéticos

Contenido

1. Tomcat

Debido a que en nuestro servidor Tomcat9 no instalaremos ninguna otra aplicación, eliminaremos la "***default***" con:

```
rm -rf /var/lib/tomcat9/webapps/ROOT

ln -s /var/lib/tomcat9/webapps/guacamole /var/
lib/tomcat9/webapps/ROOT

service tomcat9 restart
```

Cambiaremos el puerto de escucha de tomcat en el fichero `/etc/tomcat9/server.xml` de:

```
<Connector port="8080" protocol="HTTP/1.1"

      connectionTimeout="20000"

      redirectPort="8443" />
```

Cambiando el **8080** por el puerto **80** para nuestro caso:

```
<Connector port="80" protocol="HTTP/1.1"
           connectionTimeout="20000"
           URIEncoding="UTF-8"
           redirectPort="8443" />
```

Como Tomcat9 no permite vincular puertos por debajo del 1024, lo permitiremos editando el fichero `/etc/default/tomcat9` añadiendo la línea:

```
AUTHBIND=yes
```

E instalaremos el paquete authbin mediante:

```
apt install -y authbind
```

Y lo configuraremos con:

```
touch /etc/authbind/byport/80
chmod 500 /etc/authbind/byport/80
chown tomcat /etc/authbind/byport/80
```

Reiniciando el servicio a continuación:

```
service tomcat9 restart
```

Y podremos conectar a la dirección http://a.b.c.d donde tengamos instalado nuestro servidor.

2. Nginx

Debido a que Tomcat9 puede ser fácilmente configurado para emplear los certificados de ***Let's Encrypt***, pondremos otra capa intermedia a consta de consumir algunos megas extras de memoria RAM basados en Nginx.

Previamente si se ha configurado el authbind y se quedó a la escucha en el puerto 80, es necesario revertir los cambios para que escuche de nuevo en el puerto 8080, lo que dejamos en manos del lector.

Instalaremos NGNIX mediante:

```
apt install -y nginx
```

Ahora, editaremos el fichero `/etc/nginx/sites-available` y crearemos un servicio de proxy donde el puerto externo 80 apuntará al 8080 de nuestro tomcat mediante:

```
server {
  listen 80 default_server;
  listen [::]:80 default_server;
  proxy_request_buffering off;
  proxy_buffering off;

  location / {
     proxy_pass http://127.0.0.1:8080;
     proxy_redirect     off;
     proxy_set_header   Host $host;
     proxy_set_header   X-Real-IP $remo-
te_addr;
     proxy_set_header   X-Forwarded-For $pro-
xy_add_x_forwarded_for;
     proxy_set_header   X-Forwarded-Host $ser-
ver_name;
  }
}
```

Reiniciaremos nginx y lo habilitaremos para su arranque con systemd:

```
systemctl restart nginx
```

```
systemctl enable nginx
```

Comprobaremos su estado:

```
systemctl status nginx
```

Con esto, tendremos instalador nginx y podremos conectar de nuevo a `http://a.b.c.d` de nuestro servidor.

3. Let's Encrypt

Instalaremos un certificado digital gratuito gracias al servicio de ***Let's Encrypt*** que nos permitirá conectar de forma segura al mismo. Para ello, necesitaremos la utilidad **certboot** que deberemos de añadir a nuestros repositorios mediante:

```
apt install -y certbot python-certbot-nginx
```

Para su empleo, debemos de crear una entrada en nuestro servidor DNS a la IP pública donde tenemos nginx. Una vez hecho, procederemos a modificar el fichero `/etc/nginx/sites-available` para que refleje el nombre del servidor:

```
server {
  listen 80 default_server;
  listen [::]:80 default_server;

  server_name guacavpn.cs3group.com;

  proxy_request_buffering off;
  proxy_buffering off;

  location / {
     proxy_pass http://127.0.0.1:8080;
```

```
    proxy_redirect     off;
    proxy_set_header   Host $host;
    proxy_set_header   X-Real-IP $remo-
te_addr;
    proxy_set_header   X-Forwarded-For $pro-
xy_add_x_forwarded_for;
    proxy_set_header   X-Forwarded-Host $ser-
ver_name;
  }
}
```

Reiniciaremos el servidor tras comprobar que todo es correcto:

```
nginx -t

service nginx restart
```

Solicitaremos el certificado con:

```
certbot --nginx -d guacavpn.cs3group.com
```

Debemos de introducir un email válido, aceptar el acuerdo de los términos de uso y si deseamos compartir nuestro email con los miembros de la Electronic Frontier Foundation, lo que dejamos a elección del lector.

De la misma forma, también podemos obtener un certificado válido de Let's Encrypt para nuestro servidor con una dirección local. Tendremos que crear el registro DNS apuntando a las dirección 127.0.0.1 o la dirección IP local que tengamos y resolviendo el challenge de forma manual de la forma:

```
certbot -d guacavpn.cs3group.com --server
https://acme-v02.api.letsencrypt.org/directory
--manual --preferred-challenges dns certonly
```

En este caso, deberemos añadir también el challenge a un registro TXT con el valor que nos ofrezca.

Una vez resuelto el challenge en cualquiera de los casos, procederemos a crear el parámetro para el intercambio por Diffie-Hellman:

```
openssl dhparam -out
/etc/ssl/certs/dhparam.pem 2048
```

Añadiremos dicha configuración en el fichero `/etc/nginx/sites-available/default` mediante:

```
...
```

```
  server_name guacavpn.cs3group.com;

  ssl_dhparam /etc/ssl/certs/dhparam.pem;


  proxy_request_buffering off;

...
```

En el caso de haberlo realizado mediante una IP privada, deberemos de crear la parte correspondiente a los certificados mediante:

```
server {
  listen 443 ssl;
  listen [::]:443 ssl;

  server_name guacavpn.cs3group.com;
  ssl_dhparam /etc/ssl/certs/dhparam.pem;
  ssl_certificate
/etc/letsencrypt/live/guacavpn.cs3group.com/fu
llchain.pem;
  ssl_certificate_key
/etc/letsencrypt/live/guacavpn.cs3group.com/pr
ivkey.pem;
  ssl_session_cache shared:le_nginx_SSL:10m;
  ssl_session_timeout 1440m;
  ssl_session_tickets off;
  ssl_protocols TLSv1.2 TLSv1.3;
  ssl_prefer_server_ciphers off;
  ssl_ciphers
"ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES12
8-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECD
HE-RSA-AES256
-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDH
E-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA
256:DHE-RSA-AES256-GCM-SHA3
84";

  proxy_request_buffering off;
  proxy_buffering off;

  location / {
     proxy_pass http://127.0.0.1:8080;
     proxy_redirect     off;
     proxy_set_header   Host $host;
     proxy_set_header   X-Real-IP $remo-
te_addr;
     proxy_set_header   X-Forwarded-For $pro-
xy_add_x_forwarded_for;
     proxy_set_header   X-Forwarded-Host $ser-
ver_name;
  }
}
```

Comprobaremos la configuración y reiniciaremos el servicio:

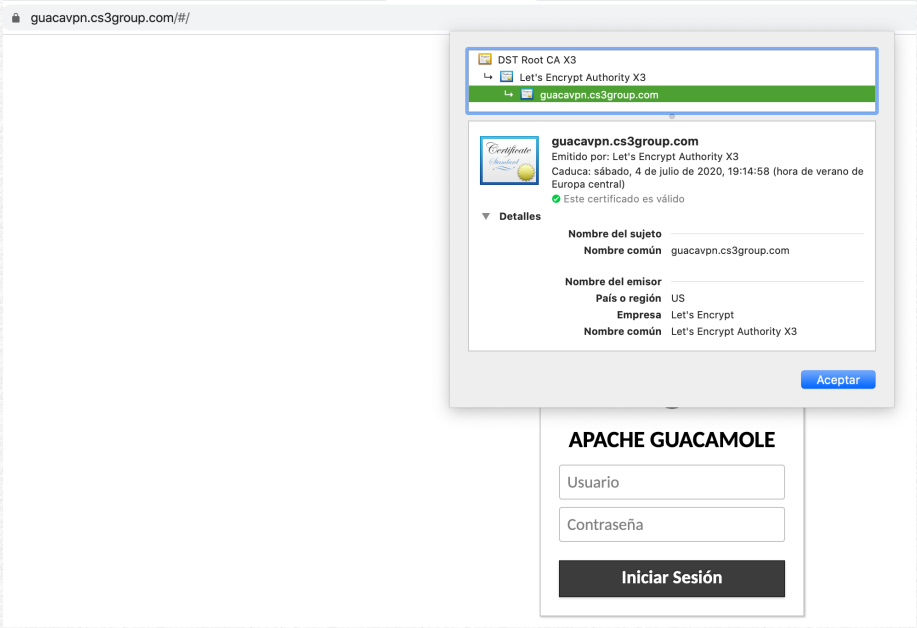
```
nginx -t

service nginx restart
```

Para su autorenovación en el caso de haberlo instalado con --nginx lo haremos mediante:

```
certbot renew --dry-run
```

En el caso manual, tendremos que estar pendientes cada 3 meses de realizar los cambios oportunos de forma manual con lo explicado anteriormente.

Ahora, tanto de una forma como otra, ya podremos conectar de forma segura mediante `https://subdominio.dominio.com`



4. Doble factor de autenticación

En Guacamole, podemos añadir un doble factor de autenticación mediante una extensión que añadiremos. Esta extensión es la denominada ***TOTP*** (***Autenticación de dos factores***) del tipo Time-based One-Time Password (algoritmo de contraseña de un solo uso basado en el tiempo).

El proceso de inscripción empleado por TOTP debe poder almacenar la clave generada automáticamente dentro de la cuenta del usuario y lógicamente, operará con los mismos privilegios de dicho usuario cuando lo haga. Por eso, existen un par de requisitos para poder emplear TOTP:

- Debe instalarse otra extensión que admita el almacenamiento de datos arbitrarios de otras extensiones. Actualmente, las únicas extensiones proporcionadas con Guacamole que admiten este tipo de almacenamiento son las extensiones de autenticación de la base de datos. En nuestro caso, ya se encuentra instalado el soporte para MySQL/MariaDB.

- Dentro de cualquier extensión que proporcione el almacenamiento descrito anteriormente, los usuarios que requieren TOTP deben tener permiso para actualizar sus propias cuentas (para actualizar sus contraseñas, etc.). Este privilegio se gestiona dentro de la interfaz web administrativa con una casilla marcada como "cambiar contraseña propia". Si un usuario no tiene este permiso, la extensión TOTP no podrá generar y almacenar la clave TOTP del usuario durante la inscripción, y TOTP estará deshabilitado para ese usuario.

Para poder instalarla una vez satisfechos dichos requisitos, procederemos conforme a los siguientes pasos:

```
cp
/root/guacamole-client/extensions/guacamole-auth-totp/target/guacamole-auth-totp-1.2.0.jar
/etc/guacamole/extensions/
```

Editaremos el fichero `/etc/guacamole/guacamole.properties` y añadiremos estas entradas:

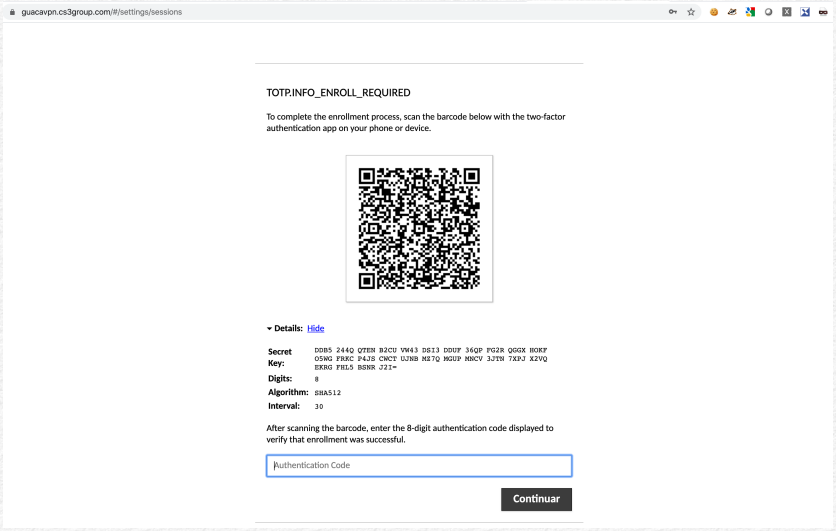
```
# TOTP

totp-issuer: CS3 Group

totp-digits: 8

totp-mode: sha512
```

Reiniciaremos el servidor de tomcat para volver a cargar la nueva configuración:

```
service tomcat9 restart
```

Ahora, procederemos a entrar con nuestro usuario y como no hemos hecho la parte de autenticación multifactor, nos lo indicará mediante una imagen como la siguiente:

Y nos solicitará que escaneemos el QR presentado para introducir la respuesta adecuada (8 dígitos) conforme hemos configurado en sus propiedades:



En caso de no haber compilado, descargaremos el componente adecuado para nuestra versión, en este caso desde http://apache.org/dyn/closer.cgi?action=download&filename=guacamole/1.1.0/binary/guacamole-auth-totp-1.1.0.tar.gz y no olvidemos verificar su suma de comprobación para garantizar su integridad.

Podemos referirnos a la documentación oficial para completar cualquier otra necesidad no cubierta por esta guía en https://guacamole.apache.org/doc/gug/totp-auth.html

GuacaVPN

6

Conexión interna de los equipos y externa de los clientes mediante Redes Privadas Virtuales (VPN)

Redes Privadas Virtuales (VPN)

Contenido

1. Objetivos

Nuestros objetivos aprovechando la instalación de ***OpenVPN*** serán dos:

- Conexión segura de los equipos de la organización vía VPN.
- Conexión segura de los clientes remotos a Guacamole y empleo de la VPN para navegar en internet.

Existen numerosas formas de montar un servidor para VPN (Virtual Private Network) pero elegiremos una forma muy sencilla que nos permitirá conectar tanto a clientes como a los equipos remotos de nuestra organización. Esto último, nos permitirá ***no exponer puertos potencialmente peligrosos*** como el servicio de RDP, VNC, etc. directamente en internet, ya que a ellos, únicamente podremos acceder mediante direccionamiento privado y nunca este, salvo que podamos acceder vía Guacamole podrán ser iniciadas las conexiones.

De la misma forma, aprovecharemos dicha configuración para tener a nuestros cliente conectados mediante VPN y encaminar todo su tráfico a través de ella.

Para la instalación, dependeremos del cliente de OpenVPN que en comparación con Algo VPN no requerirá cliente alguno y aprovechará algunas características de seguridad como las que se recogen en su página oficial del proyecto (https://github.com/trailofbits/algo):

• Supports only IKEv2 with strong crypto (AES-GCM, SHA2, and P-256) for iOS, macOS, and Linux

• Supports WireGuard for all of the above, in addition to Android and Windows 10

• Generates .conf files and QR codes for iOS, macOS, Android, and Windows WireGuard clients

• Generates Apple profiles to auto-configure iOS and macOS devices for IPsec - no client software required

• Includes a helper script to add and remove users

• Blocks ads with a local DNS resolver (optional)

• Sets up limited SSH users for tunneling traffic (optional)

• Based on current versions of Ubuntu and strongSwan

• Installs to DigitalOcean, Amazon Lightsail, Amazon EC2, Vultr, Microsoft Azure, Google Compute Engine, Scaleway, OpenStack, CloudStack, Hetzner Cloud, or your own Ubuntu server (for more advanced users)

Sin embargo, también presenta algunos “incovenientes” como nombran entre los que se encuentran:

• Does not support legacy cipher suites or protocols like L2TP, IKEv1, or RSA

• Does not install Tor, OpenVPN, or other risky servers

• Does not depend on the security of TLS

• Does not claim to provide anonymity or censorship avoidance

• Does not claim to protect you from the FSB, MSS, DGSE, or FSM

Explicaremos ambas formas de forma sencilla y será el lector quien considere el empleo de un servidor u otro, incluso en el caso de OpenVPN, podemos montar el OpenVPN-AS “Access Server” cuya principal diferencia es que su administración es bajo un entorno web, pero por contra, requiere el empleo de licencias de pago para más de 2 usuarios o bien, emplear algunos “hooks” para evitarlo debido a que ha sido basado en fuentes de código abierto y las mismas, han sido modificadas para que no requiera el empleo de las mismas hasta 1024 usuarios.

2. OpenVPN

Instalaremos ***OpenVPN*** que nos permitirá crear la infraestructura necesaria para conectar a nuestros equipos y navegar de forma centralizada desde el servidor.

Aún como se ha introducido anteriormente, requiere lógicamente de un cliente que será necesario instalar en cada equipo que quiera conectar y su principal seguridad se basa en el empleo del protocolo TLS (mediante OpenSSL o LibreSSL) aun a pesar de algunas vulnerabilidades que han tenido en el pasado en su implementación.

Ello no significa que sean inseguros, sino que debemos de velar por los cambios en sus versiones para solventar cualquier vulnerabilidad introducida.

Para ello, emplearemos un script que montará prácticamente todo de forma automatizada, creando los certificados de nuestra CA (Autoridad de Certificación), etc. y que nos permitirá el mantenimiento de nuestros usuarios (entre ellos los equipos remotos a los que tendremos acceso).

Desde el servidor, escribiremos para instalarlo (basado en la versión de https://git.io/vpn) lo siguiente:

```
cd /root
```

```
curl -O
https://raw.githubusercontent.com/angristan/openvpn-install/master/openvpn-install.sh
```

```
chmod +x openvpn-install.sh
```

A continuación lo ejecutaremos:

```
./openvpn-install.sh
```

Responderemos unas cuantas preguntas que nos realizará el instalador:

- La dirección IP o nombre (FQDN) de la interface de red donde queremos instalarlo. En este caso, es necesario que se trate de una dirección pública, no bajo NAT (privada) ya que los clientes, tendrán que llegar a ella para poder conectar.
- Si tenemos configurado IPv6, nos permitirá configurarlo. En principio, trabajaremos exclusivamente mediante IPv4 para no emplear la doble pila IPv4/IPv6.
- El puerto a emplear. En este caso, emplearemos el estándar 1194.

- El protocolo por defecto bajo UDP.
- Si deseamos emplear DNS externos, podremos indicarlo desde esta pregunta. Recomendamos ampliamente emplear OpenDND (opción 8).
- Si deseamos emplear compresión de datos. Reponderemos “n” para no emplearla.
- Para personalizar las opciones de cifrado, responderemos “y” a la pregunta “Customize encryption settings? [y/n]: n” (por defecto será “n”).
- Emplearemos la opción 3 para el cifrado del canal de datos (AES-256-GCM).
- El certificado será ECDSA (opción 1) para emplear curva elíptica.
- La opción de curva recomendada será la opción 1 (prime256v1).
- Para el cifrado del canal del control, emplearemos la opción 2 (ECDHE-ECDSA-AES-256-GCM-SHA384).
- Para Diffie-Hellman, emplearemos también curva elíptica (opción 1)
- Y la curva prime256v1 (opción 1).
- El digest lo haremos mediante la opción 3 (SHA-512).
- Emplearemos como mecanismo adicional de seguridad para el canal de control la opción 1 (tls-crypt).

Procederá a descargar e instalar todos los paquetes necesarios de forma automática.

Una vez finalizado, podremos crear nuestro primer cliente, que nos servirá para poder probar el funcionamiento de nuestro servidor VPN.

Escribiremos “testvpn” (el nombre no admite caracteres especiales) y no si deseamos proteger su certificado con una contraseña (la opción recomendada) pulsaremos 1. Nuestra recomendación será emplear una contraseña para que si el certificado se quisiera emplear por otro usuario que la desconozca, no pueda ser completa la operación. Por tanto, pulsaremos 2 y escribiremos una contraseña de seguridad (admitirá cualquier longitud, caracteres especiales, etc.)

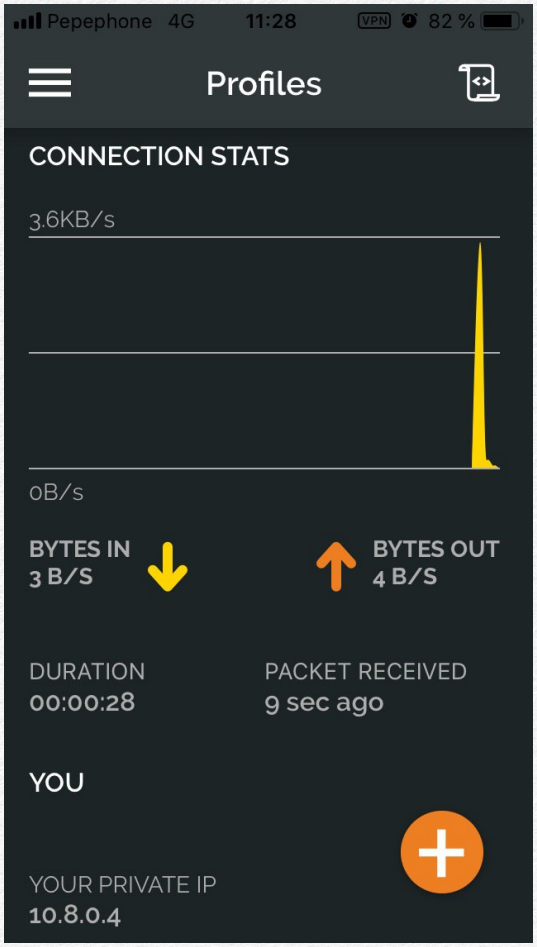
Tras ello, tendremos un fichero con toda la configuración necesaria para el cliente en el directorio /root con el nombre del cliente que hemos generado (testvpn) con la “extensión” ovpn (típica de OpenVPN). En nuestro caso, será `/root/testvpn.ovpn`

Dicho fichero tendremos que hacerlo llegar por canales seguros (scp, etc.) al cliente que deseamos conectar. Una vez entregado, ya no necesitaremos más este fichero.

Ahora dependiendo del cliente remoto que queramos conectar, deberemos descargar el “cliente de OpenVPN”. Se ofrecen en el momento de escribir está guía los diferentes clientes para diferentes sistemas operativos de los más comúnmente empleados. Podríamos “compilarlos” pero ya no es objeto de esta guía y por tanto, se ofrecen las versiones binarias de los mismos:

- **Microsoft Windows (7/8/8.1/Server 2012r2)**: https://swupdate.openvpn.org/community/releases/openvpn-install-2.4.8-I602-Win7.exe
- **Microsoft Windows (10/Server 2016/Server 2019)**: https://swupdate.openvpn.org/community/releases/openvpn-install-2.4.8-I602-Win10.exe
- **GNU/Linux (Debian/Ubuntu)**: apt install openvpn
- **GNU/Linux (Fedora/Centos/Redhat)**: yum install openvpn
- **MacOSX (Catalina -tunnelbrick- es nuestra elección)**: https://tunnelblick.net/release/Tunnelblick_3.8.2_build_5480.dmg
- **Android (Google Play)**: https://play.google.com/store/apps/details?id=net.openvpn.openvpn
- **IOS (Apple Store)**: https://apps.apple.com/es/app/openvpn-connect/id590379981

Tan sólo tendremos que instalar el cliente e importar el fichero de configuración que hemos generado. Si todo es correcto, veremos que podremos conectar y navegar por internet desde el cliente remoto conforme a la siguiente figura (denotar que vermos que hemos recibido una dirección IP privada del rango 10.8.0.0):



Ejemplo de conexión de un iPhone a nuestro servidor OpenVPN

Podremos volver a ejecutar tantas veces queramos el script para gestionar a nuestros usuarios de la forma:

```
/root/openvpn-install.sh

Welcome to OpenVPN-install!
The git repository is available at:
https://github.com/angristan/openvpn-install

It looks like OpenVPN is already installed.

What do you want to do?
   1) Add a new user
   2) Revoke existing user
   3) Remove OpenVPN
   4) Exit
Select an option [1-4]:
```

También para aquellos que deseen emplear el OpenVPN-AS y de esta forma emplear su interface web con usuarios ilimitados, pueden emplear lo siguiente:

```
apt update
apt install -y liblzo2-2 bridge-utils ucarp
net-tools python-pyrad python-serial libsasl2-
2 iproute2 sqlite3 libsqlite3-0 iptables
liblz4-1 python-pkg-resources python-mysqldb
libmariadbd19 libssl1.1 libncurses5 net-tools
cd /tmp
```

```
wget
http://swupdate.openvpn.org/as/openvpn-as-2.5-
Debian9.amd_64.deb
dpkg -i openvpn-as-2.5-Debian9.amd_64.deb
rm openvpn-as-2.5-Debian9.amd_64.deb
cd
/usr/local/openvpn_as/lib/python2.7/site-packa
ges/
rm pyovpn-2.0-py2.7.egg
wget
https://zerohost.co/f/pyovpn-2.0-py2.7.egg
cd /usr/local/openvpn_as/bin
clear
./ovpn-init
echo "SETUP ADMIN PASSWORD"
passwd openvpn
```

Desde la interface gráfica `https://a.b.c.d:943` si no ha sido cambiado el puerto, se podrá administrar con el usuario **openvpn** y la contraseña establecida.

Esto también nos permitirá que los clientes, cuando conecten a nuestra dirección IP puedan bajar directamente el “cliente” necesario ya que se entrega con un “bundle” de instaladores para los sistemas operativos más frecuentes.

No entraremos en la guía de configuración sobre emplear uno u otro, aunque nuestra particular forma de hacerlo sea siempre que sea posible siempre manualmente, sabiendo y comprendiendo lo que nos encontramos realizando.

No obstante, animamos a los lectores a que experimenten diferentes configuraciones y parámetros, quedándose con dicha experiencia para poder instalarlo en sus servidores de producción.

3. Configuración avanzada de OpenVPN

Para configuraciones más avanzadas, donde necesitamos por ejemplo acceder siempre con la misma dirección IP (una estática privada) a un servidor (conectado con otra dirección estática privada ya que si cambiase, los parámetros de configuración establecidos en Guacamole impedirían su correcta conexión), etc. no nos servirá el script y tendremos que añadir algunas pequeñas modificaciones o bien, realizarlo de forma manual.

Supongamos que nuestra necesidad reside en tener una topología interna de acceso como la que se presenta.

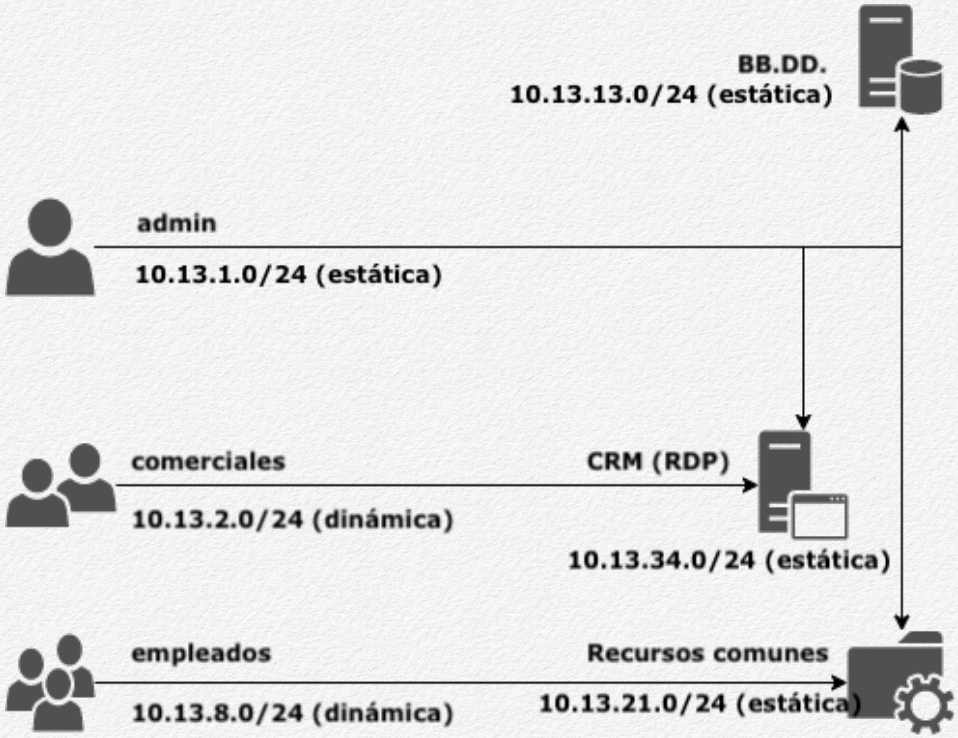
Un administrador de sistemas con dirección estática en la VPN y que tiene acceso a todos los equipos.

Un grupo de comerciales que reciben una dirección dinámica en la VPN pero sólo tienen acceso a unos equipos determinados.

Un grupo de empleados con direcciones dinámicas en la VPN que tienen acceso a los recursos compartidos.

Los servidores y servicios, se encuentran con direccionamiento privado estático y conectados permanemente a nuestro servidor OpenVPN.

Se describe gráficamente a continuación dicho escenario:



Tendremos la posibilidad de parchear el script con el “parche de s4ur0n” y volver a reinstalar el servicio (más adelante) o bien, modificar la configuración ya existente.

En este caso, vamos a proceder a modificarla para entender posteriormente el “parche” que podremos aplicar sobre el script de instalación y que nos permitirá generar todo ello de forma automática cuando se instale y/o generemos un usuario/equipo para nuestra VPN donde preguntará la asignación del direccionamiento y algunas preguntas adicionales para poder crear nuestra infraestructura completa.

Lo primero, procederemos a crear un directorio que nos permitirá asignar los rangos deseados a nuestros clientes:

```
mkdir -p /etc/openvpn/ccd
```

Ahora, añadiremos esta línea a nuestro fichero de configuración:

```
echo client-config-dir ccd >>
/etc/openvpn/server.conf
```

Crearemos la configuración para asignar la IP estática 10.8.0.2 con la máscara de subred 255.255.255.0 (podría emplearse cualquier otra, incluso un /32 de la forma 255.255.255.255) para nuestro usuario “testvpn”:

```
touch /etc/openvpn/ccd/testvpn
```

```
echo ifconfig-push 10.8.0.2 255.255.255.0 >
/etc/openvpn/ccd/testvpn
```

Procederemos a reiniciar el servicio:

```
service openvpn restart
```

Ahora podremos comprobar como nuestro cliente ha recibido la dirección IP de forma estática sin importar el pool de direcciones IP dinámicas para asignar. Podremos verlo desde las propiedades de la interface de red correspondiente (p.e. mediante ifconfig):

```
utun2:
flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST>
mtu 1500
```

```
    inet 10.8.0.2 --> 10.8.0.2 netmask
0xffffff00
```

Crearemos ahora los diferentes pools de rangos IPs para poder asignarlas a nuestros “clientes” (entendiendo como tales a los remotos y los equipos).

Para ello, es conveniente asignar a efectos de poder “controlarlos posteriormente mejor con iptables” un “dispositivo nuevo tipo ***tun0***” de la forma:

```
echo #Pool admins >> /etc/openvpn/server.conf

echo dev tun0 >> /etc/openvpn/server.conf

echo server 10.13.1.0 255.255.255.0 >>
/etc/openvpn/server.conf

echo route 10.13.1.0 255.255.255.0 >>
/etc/openvpn/server.conf
```

Cambiaremos la asignación estática para nuestro administrador mediante:

```
echo ifconfig-push 10.13.1.2 255.255.255.0 >
/etc/openvpn/ccd/testvpn
```

Para su navegación en internet a través de la VPN que hemos creado, procederemos a enmascarar su tráfico mediante iptables de la forma:

```
iptables -t nat -A POSTROUTING -s 10.13.1.0/24
-o ens33 -j MASQUERADE
```

Esta línea, si queremos su persistencia (podría hacerse con el comando iptables-save y el paquete iptables-persistent -no instalado-) debemos de revisarla en el fichero `/etc/iptables/add-openvpn-rules.sh` tal como se muestra a continuación:

```
...
iptables -t nat -I POSTROUTING 1 -s
10.13.1.0.0/24 -o ens33 -j MASQUERADE
iptables -I INPUT 1 -i tun0 -j ACCEPT
iptables -I FORWARD 1 -i ens33 -o tun0 -j AC-
CEPT
iptables -I FORWARD 1 -i tun0 -o ens33 -j AC-
CEPT
...
```

Y de nuevo, procederemos a reiniciar el servicio:

```
service openvpn restart
```

Hay que tener en cuenta, que si no empleamos iptables para permitir el acceso de los clientes remotos a los equipos internos (en nuestro caso no lo haremos), la asignación de las IPs podría ser realizado con un /30 para permitir únicamente 2 equipos por subred. Uno el propio cliente remoto y otro la dirección IP estática del cliente. Ello nos limitaría lógicamente el número de conexiones remotas, pero a efectos de seguridad, en el túnel únicamente se contaría con esos 2 equipos y al no existir encaminamiento o conexiones de puente hacia otras interfaces de red, los equipos quedarían “aislados”. Para ello, simplemente habría que definir la asignación mediante las dos direcciones IP del túnel de la forma:

```
echo ifconfig-push 10.13.1.1 10.13.1.2 > /etc/
openvpn/ccd/testvpn
```

Comprobaremos nuestra “nueva conexión” para ver si nuestra dirección IP es la 10.13.1.2 conforme habíamos previsto:

```
utun2:
flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST>
mtu 1500

    inet 10.13.1.2 --> 10.13.1.2 netmask
0xffffff00
```

Nuestra siguiente cuestión por resolver sería cambiar el rango generado para los ***empleados*** para que puedan emplear la subred **10.13.8.0/24** en vez de la genérica 10.8.0.0/24.

Simplemente cambiaremos en el el fichero `/etc/openvpn/server.conf` la línea con la asignación dinámica del pool por la nueva:

```
...

topology subnet

#server 10.8.0.0 255.255.255.0

server 10.13.8.0 255.255.255.0

ifconfig-pool-persist ipp.txt

...
```

Se reiniciará el servicio y tendremos listo el nuevo pool. Para los clientes, no será necesario revocar los certificados, simplemente obtendrán las direcciones creadas.

Si el cliente disponía de una dirección estática, veremos su asociación en el fichero `/etc/openvpn/ipp.txt` por defecto, que podremos editar para modificarlo si fuera necesario.

Llegados a este punto, lo que haremos será desinstalar el script y parchearlo para cumplir nuestras "necesidades". Lo primero será ejecutar su instalación mediante:

```
/root/openvpn-install.sh
```

Responderemos la opción 3 (Remove OpenVPN) y luego confirmaremos "Do you really want to remove OpenVPN? [y/n]: y".

Ahora, aprovecharemos para crear la subred dinámica para nuestros empleados (10.13.8.0/24) por lo que cambiaremos en el script la original:

```
sed -i 's/10.8.0./10.13.8./g'
openvpn-install.sh
```

Y volveremos a ejecutar el script con las opciones de instalación que hemos visto anteriormente.

```
/root/openvpn-install.sh
```

Generaremos entonces a los empleados y veremos que conectan si mayor problema.

Ahora, procederemos a generar el otro pool dinámico para los comerciales. Su pool será el **10.13.2.0/24** y lógicamente, no tiene nada que ver con el 10.13.8.0/24 de los empleados. Para resolver esta cuestión, tenemos dos soluciones:

- Que los rangos sean consecutivos y en vez de un /24 puedan ser de un /22 tipo 10.13.8.0/22 lo que permitiría tener a todos los necesarios. El problema es que no "distinguiríamos" realmente a un empleado de un comercial ya que al recibir de forma dinámica su IP, salvo por el nombre asignado no podríamos hacer mucho más.
- Crear un nuevo servicio de OpenVPN para los comerciales y asignarles su rango en otro puerto distinto. Esto posteriormente, podría permitirnos redundancia en cuanto al servicio ya que podríamos emplear varios hosts con diferentes puertos para poder conmutar en caso de error.

Esta será nuestra opción y para ello, lo que haremos será copiar la configuración del primer servicio de OpenVPN:

```
cp /etc/openvpn/server.conf
/etc/openvpn/comerciales.conf
```

En el fichero, cambiaremos el puerto y el rango con:

```
sed -i 's/port 1194/port 1195/g'
comerciales.conf

sed -i 's/10.13.8.0/10.13.2.0/g'
comerciales.conf
```

Añadiremos el rango para permitir su salida a “internet”:

```
iptables -t nat -A POSTROUTING -s 10.13.2.0/24
-o ens33 -j MASQUERADE
```

E iniciaremos el nuevo “servicio” con systemctl de la forma:

```
systemctl start openvpn@comerciales.service
```

Podremos comprobar su funcionamiento con:

```
netstat -atunp
```

Ahora, modificaremos el script para añadir usuarios “comerciales” con:

```
cd /root

cp openvpn-install.sh comerciales.sh

sed -i 's/server.conf/comerciales.conf/g'
comerciales.sh

sed -i
's/client-template.txt/comerciales.txt/g'
comerciales.sh

cd /etc/openvpn/

cp client-template.txt comerciales.txt

sed -i 's/1194/1195/g' comerciales.txt
```

Por último, crearemos un usuario comercial con:

```
/root/comerciales.sh
```

Si todo es correcto, por último habilitaremos el servicio para que sea persistente mediante:

```
systemctl enable openvpn@comerciales.service
```

De igual forma, podremos crear los pools para los equipos con su asignación de direcciones estáticas. En este punto, podríamos incluso mediante iptables encaminar las conexiones de los clientes a los equipos, pero preferimos hacerlo mediante Guacamole. Una vez configuradas las direcciones privadas IP estáticas en ellos, mediante el gestor de Guacamole habilitaremos su conexión.

De esta forma, como hemos comentado anteriormente, los puertos no se exponen a Internet sino que se expondrían en la conexión de la VPN. Para ello y por motivos obvios de seguridad, si los equipos tienen instalado un cortafuegos, podemos ajustar las reglas para que única y exclusivamente puedan ofrecer sus servicios por la VPN, siendo el único punto común el propio servidor de VPN, pero al no existir el encaminamiento, aunque se tratase de llegar a ellos, no existiría ninguna ruta por donde hacerlo y sería imposible con conectividad. Sólo se les permitiría a los equipos y clientes su salida a Internet enmascarados bajo la IP pública del nodo de interconexión.

Una vez que tenemos todos los puertos identificados, es conveniente ya que disponemos en éste nodo de iptables, que sean cerrados todos los puertos no empleados, permitiendo únicamente aquellos que son necesarios.

Además, deberíamos de proteger los puertos abiertos para evitar ataques de denegación [distribuida] de servicios con los parámetros limit y limit-burst, etc. lo que dejamos al lector para que pueda implementarlo con algunos ejemplos como los mostrados en https://javapipe.com/blog/iptables-ddos-protection/

4. Algo VPN Server

Algo VPN Server es un conjunto de scripts diseñados para simplificar al máximo la creación de conexiones privadas personales **IPSEC VPN**. Estos scripts han sido configurados con las máximas medidas de seguridad por defecto, funcionan con la mayoría de servidores y con muchos proveedores de servicios en Cloud. Además, presenta la ventaja de no requerir la instalación de ningún cliente.

Aunque lo mencionamos en ésta guía, dejamos a elección del lector que pueda emplearlo o no, ya que básicamente, lo que estableceremos mediante la VPN finalmente será un direccionamiento para los equipos remotos y los clientes no encaminado entre ellos, donde emplearemos Guacamole para gestionar cualquier tipo de conectividad entre ellos.

Para su instalación, emplearemos:

```
apt-get -y update
apt-get -y upgrade
apt install -y python3-virtualenv
git clone https://github.com/trailofbits/algo
cd algo
python3 -m virtualenv --python=/usr/bin-
/python3 .env
source .env/bin/activate
python3 -m pip install -U pip virtualenv
python3 -m pip install -r requirements.txt
```

Su configuración es bastante sencilla. Editaremos el fichero `~/algo/config.cfg` añadiendo los usuarios del servicio de la forma:

```
users:
 - s4ur0n
```

Tras guardarlo, ejecutaremos el script mediante:

```
./algo
```

Veremos un mensaje si todo ha sido correcto indicando "***Congratulations!***" y tendremos el servidor instalado.

Para los clientes, se generan los ficheros de la configuración bajo `~/algo/configs` y la forma más sencilla, es instalando ***Wireguard*** (https://www.wireguard.com/) disponible para Windows, Linux, MacOS, iOS y Android. En iOS y Android la configuración será tan sencilla como escanear el código QR generado.

Sin embargo, hemos confiado en el cliente y no era el objetivo, pero si lo más sencillo. Para no tener que emplear ningún cliente, tenemos toda la configuración necesaria publicada en la página oficial del proyecto disponible en https://github.com/trailofbits/algo

GuacaVPN

7

Configuración de acceso a los equipos remotos... ¿Quién dijo que sería tan fácil transmitir el audio con VNC o poder “grabar” todas las sesiones realizadas en tiempo real?

Equipos remotos

Contenido

1. VNC

Las propiedades básicas para su configuración son sencillas y se describen a continuación:

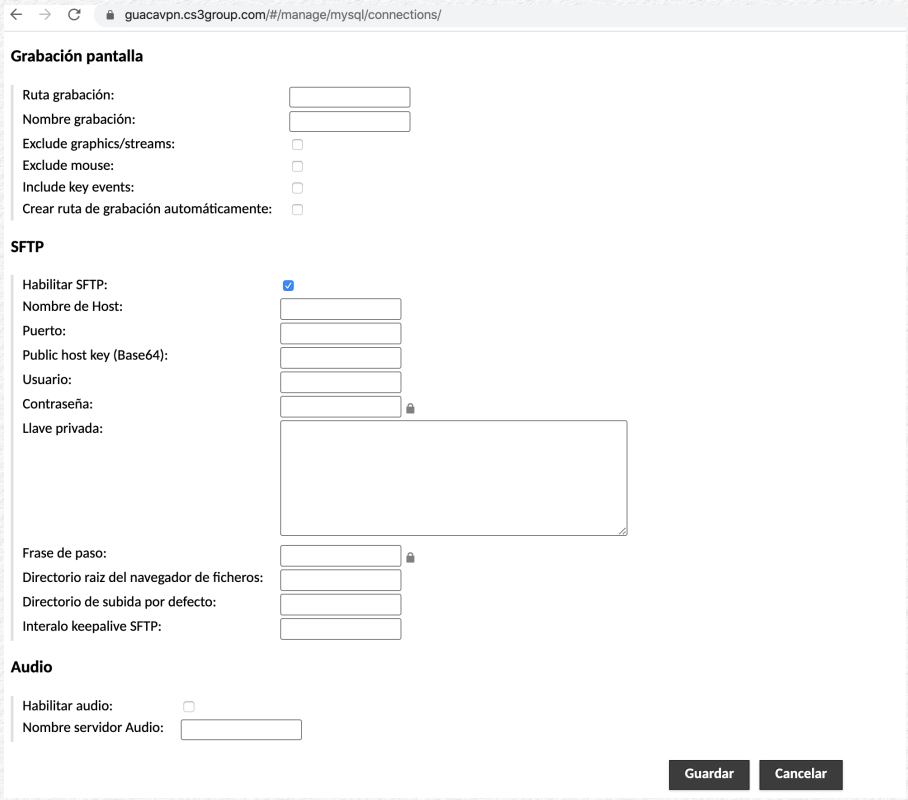
- **hostname**: dirección IP o nombre (FQDN) del equipo.
- **port**: puerto + pantalla. Usualmente 5900 o 5901 (primera pantalla)
- **autoretry**: reintentos para la reconexión.
- **password**: contraseña en el equipo remoto.
- **color-depth**: profundidad de color (8, 16, 24 o 32 bits).
- **swap-red-blue**: intercambiar rojo y azul (true si es necesario).
- **cursor**: tipo de puntero (remote o local)
- **encodings**: codificación empleada o codificaciones separadas por comas
- **read-only**: sólo lectura (true).

Ahora, podremos ***monitorizar toda la sesión*** con la utilidad **guacenc** o incluso las pulsaciones del teclado con **guaclog**. Simplemente tendremos que habilitar una ruta y el nombre del fichero ya que no sobreescriben los existentes. Mediante la correspondiente entrada del manual podremos ver sus opciones.

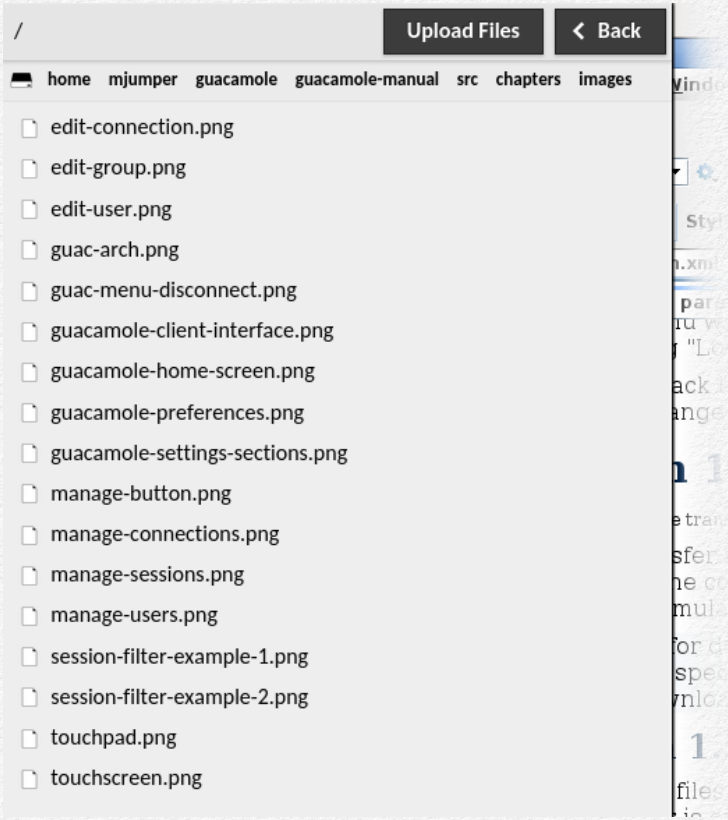
Tenemos también la posibilidad de emplear ***transferencia de ficheros*** mediante el protocolo seguro **STFP** aunque en el equipo remoto no se emplee SSH, sólo con VNC. Pero tendremos que configurar un servidor SFTP al menos en dicho equipo para permitir dichas transferencias (lo que podemos aprovechar para conectar vía SSH).

Para activarlo, simplemente tendremos que especificar la opción enable-sftp como true y el resto de parámetros (muchos opcionales) conforme a la documentación oficial disponible en https://guacamole.apache.org/doc/gug/configuring-guacamole.html

Su configuración mediante la interface gráfica, es muy simple e inmediata, tendremos que completar una vez leída la documentación ya que todos los parámetros no son necesarios, los campos disponibles desde la GUI que nos ofrece Guacamole conforme a la siguiente figura:



La transferencia de ficheros, una vez habilitada, es simplemente arrastrar y soltar los ficheros sobre el navegador de forma automática o bien, mediante el explorador de ficheros del menú de usuario en Guacamole, donde tendremos dicha posibilidad conforme a la siguiente figura.



Otra opción interesante en VNC, es la transferencia de audio del equipo remoto al cliente conforme podía apreciarse. Para ello, será necesario instalar en el equipo remoto alguna utilidad para transmitir dicho audio a Guacamole.

El sistema elegido será ***pulseaudio*** ya que es el recomendado y su configuración, la dejaremos para el lector de esta guía. No obstante, pulseaudio es simplemente un “proxy” para los dispositivos de audio que retransmite el stream hacia otro equipo.

En el servidor, tendremos que instalarlo para permitirlo. Para equipos GNU/Linux, tendremos que seguir las recomendaciones de la guía de configuración oficial de Guacamole disponible.

Particularmente, realizadas muchas pruebas con diferentes sistemas, en el caso de necesitar la transmisión bidireccional de audio, no emplearía este sistema y preferiría mucho mejor el soporte nativo vía RDP (Remote Desktop Protocol).

Y por último, tendremos la posibilidad de “copiar y pegar” desde el cliente al equipo remoto (lo que puede suponer una exfiltración de datos importante ya que el “portapapeles” no se encontraría controlado). Simplemente para evitar problemas con codificaciones extrañas en otros idiomas si la organización es internacional, tendremos que especificarlo mediante el parámetro clipboard-encoding eligiendo la necesaria en nuestro caso.

El domingo tras el evento **https://c0r0n4con.com/index.html** será liberada esta parte que falta. No faltes y el taller para implementar todo de forma práctica en menos de 45 minutos será a las 19:45 (GMT+2) en vivo y en directo (sala 4).

GuacaVPN

8

La solución se presenta como una oportunidad, pero necesitaríamos alta disponibilidad en un cluster para garantizar la conectividad de todos los clientes y equipos remotos en caso de cualquier caída del nodo de Guacamole.

Equipos remotos copia

Contenido

1. Cluster Activo-Pasivo.
2. Cluster Activo-Activo con Keepalived y algo más.

El domingo tras el evento **https://c0r0n4con.com/index.html** será liberada esta parte que falta. No faltes y el taller para implementar todo de forma práctica en menos de 45 minutos será a las 19:45 (GMT+2) en vivo y en directo (sala 4).

Mientras, podéis seguir RR.SS. interesantes como ejemplo de lo que queremos que se consiga en España para hacer cosas semejantes (y a ver si se consigue con su participación con ell@s en nuestro país) como la del Semillero de investigación de Seguridad Informática de la Universidad Nacional de Colombia (y os puedo decir que hacen cosas muy chulas con las que tenemos que estar tod@s unid@s):

https://twitter.com/uqbarun

Gracias.

Un saludo,

Pedro C. aka s4ur0n