Crear Certificado SSL En Apache Para Ubuntu

Como Crear Un Certificado SSL En Apache Para Ubuntu 14.04

Secure Sockets Layer ó SSL, es un protocolo de seguridad creado con la finalidad de colocar el tráfico normal en un trafico seguro, esto, mediante la protección de un fuerte encriptación.

Este protocolo le permite que el tráfico se envíe de forma segura entre las partes remotas sin la posibilidad de que el tráfico sea interceptado y leído por alguien en el medio. También juega un papel decisivo en la validación de la identidad de los dominios y servidores a través de Internet mediante la creación de un servidor como de confianza y genuina por una autoridad certificadora.

En esta guía, vamos a mostrar cómo **crear un certificado SSL** autofirmado para Apache en un **servidor de Ubuntu 14.04**, lo que le permitirá encriptar el tráfico a su servidor. Si bien esto no proporciona el beneficio de la validación de terceros de la identidad de su servidor, que cumple los requisitos de aquellos que desean simplemente para transferir información de forma segura.

Requisitos Previos

Estaremos operando como un usuario no root con privilegios sudo en esta guía. También vas a necesitar tener instalado Apache. Si no tienes esto en funcionamiento, se puede arreglar rápidamente escribiendo:

[bash]
sudo apt-get update
sudo apt-get install apache2
[/bash]

Paso Uno – Activar El Módulo SSL

El soporte SSL en realidad viene como paquete estándar del **Apache en Ubuntu 14.04**. Simplemente necesitamos habilitarlo para poder usar el SSL en nuestro servidor. **Habilita el módulo** escribiendo en la consola:

[bash]

sudo a2enmod ssl

[/bash]

Después de habilitar el **SSL**, tendrá que reiniciar el servidor web para que el cambio sea reconocido:

[bash]

sudo service apache2 restart

[/bash]

Con esto, nuestro servidor web es ahora capaz de manejar SSL si lo configuramos correctamente para hacerlo.

Paso Dos – Crear Un Certificado SSL Autofirmado

Vamos a empezar por la creación de un subdirectorio dentro de las carpetas de configuración de Apache para colocar los archivos de certificado que vamos a estar creando:

[bash]

sudo mkdir /etc/apache2/ssl

[/bash]

Ahora que tenemos un lugar para colocar la llave y el certificado, podemos crear los dos en un solo paso escribiendo lo siguiente en la consola:

[bash]

sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/apache2/ssl/apache.key -out /etc/apache2/ssl/apache.crt

[/bash]

Los detalles de la configuración son los siguientes:

- **openssl** : Se trata de la herramienta de línea de comandos básica proporcionada por OpenSSL para crear y administrar certificados, llaves, solicitudes de firma , etc
- **req** : Esto especifica un sub comando para la solicitud de certificate signing request X.509 (CSR). X.509 es un estándar de infraestructura de clave pública que SSL se añade por su clave y certificado administrado. Dado que estamos queriendo crear un nuevo certificado X.509.
- **x509**: Esta opción especifica que queremos hacer un archivo de certificado auto firmado en lugar de generar una solicitud de certificado.
- **nodes**: Esta opción le dice a OpenSSL que no queremos asegurar nuestro archivo de clave con una contraseña. Tener un archivo con clave protegida por contraseña haría que **Apache se inicie automáticamente**, ya que habría que introducir la contraseña cada vez que

se reinicia el servicio.

- **days 365**: Esto especifica que el certificado que estamos creando será válida por un año.

- **newkey rsa:2048**: Esta opción creará la solicitud de certificado y una clave privada nueva, al mismo tiempo. Esto es necesario ya que nosotros no creamos una clave privada con antelación. El **rsa: 2048** le dice a OpenSSL que genere una clave **RSA** que es de **2048 bits de longitud**.
- **keyout**: los nombres de este parámetro, es del archivo de salida para el archivo de clave privada que se está creando.

- **out**: Esta opción da nombre al archivo de salida para el certificado que estamos generando.

Cuando se pulse «Enter», se le pedirá una serie de preguntas.

El punto más importante que se solicita es la línea que dice «Nombre común (por ejemplo, el FQDN del servidor ó su nombre)». Debe introducir el nombre de dominio que desea asociar con el certificado, o la dirección IP pública del servidor, si no cuenta con un nombre de dominio.

Lo anterior se veria algo como lo siguiente:

[bash]

```
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:New York
Locality Name (eg, city) []:New York City
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Your Company
Organizational Unit Name (eg, section) []:Department of Kittens
Common Name (e.g. server FQDN or YOUR name) []:your_domain.com
Email Address []:your_email@domain.com
```

[/bash]

Tercer Paso – Configurar Apache Para Usar SSL

Ahora que ya tenemos nuestro certificado y la clave disponibles, podemos configurar Apache para que pueda utilizar estos archivos en un archivo de host virtual. Usted puede aprender más acerca de cómo configurar hosts virtuales Apache aquí.

En lugar de basar nuestro archivo de configuración en el archivo **000-default.conf** en el subdirectorio de las páginas web disponibles, vamos a basar esta configuración en el archivo **default-ssl.conf** que contiene un poco de la configuración SSL de forma predeterminada.

Abra el archivo con privilegios de root:

[bash]sudo nano /etc/apache2/sites-available/default-ssl.conf[/bash]

Con los comentarios eliminados, el archivo se veria algo como esto:

[bash]

```
<IfModule mod_ssl.c>
<VirtualHost _default_:443>
ServerAdmin webmaster@localhost
DocumentRoot /var/www/html
ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined
SSLEngine on
SSLCertificateFile /etc/ssl/certs/ssl-cert-snakeoil.pem
SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key
<FilesMatch "\.(cgi|shtml|phtml|php)$">
SSLOptions +StdEnvVars
```

```
</FilesMatch>
<Directory /usr/lib/cgi-bin>
SSLOptions +StdEnvVars
</Directory>
BrowserMatch "MSIE [2-6]" \
nokeepalive ssl-unclean-shutdown \
downgrade-1.0 force-response-1.0
BrowserMatch "MSIE [17-9]" ssl-unclean-shutdown
</VirtualHost>
</IfModule>
```

[/bash]

Esto puede parecer un poco complicado, pero por suerte, no es necesario que te preocupes por la mayoría de las opciones aquí.

Queremos dar la configuración básica para un host virtual (ServerAdmin, ServerName, ServerAlias□□, DocumentRoot, etc), así como cambiar la ubicación donde Apache buscara el certificado SSL y la llave.

Al final, se verá algo como esto. Las entradas en rojo son las modificaciones que se hicieron en el archivo original:

[bash]

```
<IfModule mod_ssl.c>
<VirtualHost _default_:443>
ServerAdmin admin@example.com
ServerName your_domain.com
ServerAlias www.your_domain.com
DocumentRoot /var/www/html
ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined
SSLEngine on
SSLCertificateFile /etc/apache2/ssl/apache.crt
SSLCertificateKeyFile /etc/apache2/ssl/apache.key
<FilesMatch "\.(cgi|shtml|phtml|php)$">
```

```
SSLOptions +StdEnvVars
</FilesMatch>
<Directory /usr/lib/cgi-bin>
SSLOptions +StdEnvVars
</Directory>
BrowserMatch "MSIE [2-6]" \
nokeepalive ssl-unclean-shutdown \
downgrade-1.0 force-response-1.0
BrowserMatch "MSIE [17-9]" ssl-unclean-shutdown
</VirtualHost>
</IfModule>
```

[/bash]

Guarde y salga del archivo cuando haya terminado.

Paso Cuatro – Active El Host Virtual SSL

Ahora que ya hemos configurado nuestro host virtual, debemos de habilitar el SSL. Podemos hacer esto escribiendo lo siguiente en la consola:

[bash]

```
sudo a2ensite default-ssl.conf
```

[/bash]

Entonces, necesitamos reiniciar Apache nuevamente para cargar el nuevo archivo del host virtual:

[bash]

```
sudo service apache2 restart
```

[/bash]

Esto debería permitir que su nuevo host virtual, servirá de

contenido cifrado utilizando el certificado SSL que ha creado.

Quinto paso – Comprobación De La Configuración

Ahora que ha preparado todo, usted puede probar la configuración visitando el nombre de dominio de su servidor o la dirección IP pública después de especificar el protocolo https://, así:

[bash]

https://nombre_dominio_ó_ip_servidor

[/bash]

Usted recibirá un aviso de que el navegador no puede verificar la identidad de su servidor, ya que no ha sido firmado por una de las autoridades de certificación de confianza.

Esto se esperaba, ya que tenemos auto firmado nuestro certificado. Mientras que nuestro certificado no sea «valido» para nuestros usuarios, ya que no ha tenido ninguna interacción con una autoridad de certificación de confianza, todavía será capaz de cifrar la comunicación.

Dado que se espera, usted puede golpear el botón o lo que sea semejante opción que tienes en tu navegador «**Proceed anyway**«.

Una vez que sigamos a nuestro sitio web, podemos ver el contenido que deseemos mostrar, pero este estara cifrado ;) Puede comprobarlo haciendo clic en el icono del candado en la barra de menú:

Se puede ver en la sección verde central que la **conexión está**

cifrada.

Conclusión

Usted debe tener ahora el **SSL habilitado en su sitio web**. Esto ayudará a proteger la comunicación entre visitantes y su sitio, pero va a advertir a cada usuario que el navegador no puede verificar la validez del certificado.

Si está pensando en lanzar un sitio público y seguro para el usuario, tendrá la necesidad de un SSL, será mejor comprar un certificado SSL respaldado para que se muestre como un sitio bastante seguro.