Seguridad Web

Transcribed on July 10, 2025 at 3:16 PM by Minutes AI

Speaker 1 (00:02)

Bienvenidos a esta nueva sesión.

En esta sesión vamos a hablar sobre los protocolos SSL y TLS.

Vamos a ver cómo funcionan y cómo se pueden generar certificados autofirmados con la herramienta OpenSSL.

Tras esto, realizaremos un ejemplo de montaje de estos certificados en un servidor web para comparar el tráfico con y sin certificado.

Comenzamos hablando sobre un aspecto fundamental de la seguridad en línea, que son los protocolos SSL y, actualmente, TLS.

En un mundo donde la comunicación a través de Internet es omnipresente, la protección de nuestros datos se ha convertido en una prioridad absoluta.

Ssl, que es Secure Socket Layer, y su sucesor TLS de Transport Layer Security son los pilares de esta seguridad en la web y entender cómo funcionan es claro.

Clave Presente la protección de nuestros datos se ha convertido en una prioridad absoluta.

Ssl, que es Secure Socket Layer, y su sucesor TLS de Transport Layer Security, son los pilares de esta seguridad en la web y entender cómo funcionan es clave para comprender cómo proteger nuestras comunicaciones en línea.

Para comprender cómo TLS garantiza esta seguridad, primero debemos sumergirnos en su funcionamiento.

Y es que en primer lugar distinguimos el saludo o handshake y a partir de ahí moviliza la concesión segura de la información.

En el handsake, las partes involucradas en la comunicación, normalmente el cliente y el servidor, se autentican mutuamente y acuerdan una clave de sesión segura.

Esto es esencial para garantizar que las partes sean quienes dicen ser y para establecer un canal seguro de comunicación.

Durante este proceso también se negocian los parámetros de cifrado y autenticación que se utilizarán para proteger la transferencia de datos.

Una vez completado el proceso de saludo o el fanse, entramos en la transferencia segura de la información.

Aquí, los datos transmitidos entre el cliente y el servidor se cifran utilizando la clave de sesión acordada, lo que garantiza que sólo las partes autorizadas puedan acceder a ello.

Además, se aplican medidas para asegurar la integridad de los datos, como funciones hash y códigos de autenticación de mensajes que nos permiten verificar que los datos no han sido alterados durante la transmisión.

La importancia de TLS no puede ser exagerada.

Este protocolo es fundamental para proteger nuestras comunicaciones en línea, desde transacciones financieras hasta intercambio de correos electrónicos y datos sensibles.

Proporciona una capa de seguridad mundial que nos permite confiar en que nuestras interacciones en la web están protegidas contra amenazas como el robo de datos y el espionaje en la red.

En resumen, podemos decir que SSL y TLS es el cimiento sobre el cual se construye la seguridad en línea.

Su funcionamiento robusto y su capacidad para proteger nuestras comunicaciones son esenciales en un mundo cada vez más conectado.

Vamos a pasar a continuación a ver cómo podemos generar los certificados necesarios para establecer un túnel TLS y que nuestra información viaje cifrada entre el cliente y el servidor.

Para ello haremos uso de la herramienta OpenSSL, que es una herramienta de código abierto ampliamente utilizada en la seguridad digital, sobre todo para trabajar con este tipo de certificados y protocolos, como su propio nombre indica, SSL y TLS.

Vamos a comenzar por la generación de certificados autofirmados utilizando esta herramienta.

Un certificado autofirmado es un tipo de certificado digital en el que la entidad emisora y la entidad propietaria son la misma.

Esto significa que el propietario del certificado se valida a sí mismo en lugar de depender de una entidad de certificación externa.

Con OpenSSL podemos generar fácilmente un certificado autofirmado con un simple comando.

Este proceso implica la creación de un par de claves, pública y privada, y luego la generación de un certificado firmado con la clave privada.

Más adelante podremos juntar este par de archivos generados en un único archivo PKCS 12 o bien conocido como P.

En cuanto a las ventajas que esto ofrece, se puede decir que la exoneración de 11 autofirmado es un proceso rápido y sencillo que no requiere la intervención de una autoridad de certificación externa.

Esto significa que podemos implementar rápidamente certificados seguros en nuestros sistemas sin pasar por un largo proceso de validación.

Debido a esto, los certificados autofirmados son especialmente útiles en entornos de desarrollo y en entornos de prueba, donde la seguridad es importante pero no es necesario un certificado emitido por una entidad de certificación reconocida.

Esto nos permite probar y desarrollar nuestras aplicaciones de manera segura, sin tener que preocuparnos por adquirir y mantener certificados válidos.

Por último, al generar certificados autofirmados mantenemos el control total sobre la emisión y la validez del certificado.

No estamos limitados por las políticas o los tiempos de respuesta de una autoridad externa, lo que nos brinda una mayor independencia y flexibilidad en la gestión de la seguridad de nuestros sistemas.

El comando que vamos a utilizar para la generación del certificado autofirmado es openssl rec x también como parámetro newkey con el valor, por ejemplo, RSA 2048, otro parámetro keyout con el nombre del fichero que lo queramos guardar, otro que sería out con el nombre del fichero que queramos guardar y por último uno con los días, el parámetro days.

Pero vamos a verlo por partes, desglosando cada uno de estos argumentos.

En primer lugar tenemos el comando principal que invoca la herramienta openSSL.

Después, con rec lo que tenemos es la operación que se va a realizar, que es una solicitud de certificado.

En este caso se está solicitando la creación de un certificado autofirma.

Luego tenemos el parámetro x, que este parámetro especifica que el certificado generado será un certificado x, que es un estándar ampliamente utilizado para los certificados digitales.

El siguiente argumento con Newkey lo que estamos haciendo es indicar que tenemos que generar un nuevo par de claves, en este caso RSA de 2048 bits.

Rsa es un algoritmo de cifrado ampliamente utilizado para la generación de claves públicas y privadas.

El siguiente parámetro que tenemos sería keyout, seguido del nombre de archivo de salida donde se guardará la clave privada generada.

En este caso se le está definiendo que se guardará un archivo key pen, es decir que este va a ser la extensión o el tipo de certificado que vamos a tener, un punto pE.

Otro parámetro que tendríamos sería out para especificar el nombre del archivo de salida donde se guardará el certificado generado.

En este caso lo guardamos en cert pe.

Aquí ya tendríamos los dos ficheros, tanto la clave como el certificado generado por RSA.

Por último, el argumento Days, vamos a indicar la cantidad de días durante durante los cuales este certificado será válido, en este caso vamos a fijarlo por un año o lo que es lo mismo 365 días.

En resumen y como se puede ver, este comando genera un par de claves pública y privada RSA de 2048 bits y luego utiliza estas claves para crear un certificado autofirmado que será válido durante un año.

El certificado resultante se guarda en un archivo llamado C y la clave privada correspondiente se guardará en un archivo llamado key PEM.

Por otro lado, vamos a comentar la conversión de certificados digitales, específicamente sobre la transformación de un certificado en formato PEM como el que acabamos de generar, a otro en formato P.

Antes de entrar en detalles sobre la conversión, es importante entender qué es un certificado en formato PEM.

Pem, que es precisamente privacy en access mail, es un formato de archivo comúnmente utilizado para almacenar certificados digitales y claves privadas en la web.

Estos archivos están codificados en texto ASCII, lo que los hace legibles y editables por los humanos.

¿Entonces, por qué considerar la conversión de un certificado formato Pem a otro en formato b?

Las ventajas son varias y significativas.

En primer lugar, la portabilidad compatibilidad los archivos p, también conocidos como archivos Tkcs o archivos de contenedor de Cle, son altamente portátiles y compatibles con una amplia gama de sistemas y aplicaciones.

Eso significa que podemos utilizar un certificado convertido en P en una variedad de plataformas y entornos sin problemas de compatibilidad.

Por otro lado, los archivos p tienen la capacidad única de almacenar no sólo el certificado digital, sino también la clave privada asociada y opcionalmente, certificados de cadena de confianza.

Esto significa que podemos proteger la clave privada y el certificado en un solo archivo utilizando una contraseña fuerte para asegurar su acceso.

Esto reduce significativamente el riesgo de comprometer la clave privada y garantiza una mayor seguridad en nuestras comunicaciones en línea.

Por último, al tener el certificado digital y la clave almacenados juntos en un solo archivo, la gestión y el mantenimiento se vuelven mucho más simples y eficientes.

No es necesario rastrear múltiples archivos PEN.

En su lugar, tenemos todo lo necesario en un único archivo p, lo que facilita el proceso de respaldo, transferencia y configuración de certificados en diferentes sistemas y entornos.

En resumen, la conversión de un certificado en formato pem a otro formato p ofrece una serie de ventajas significativas en términos de portabilidad, seguridad y facilidad de gestión.

Y ahora que sabemos esto, vamos a ver el comando utilizado para convertir un certificado 1 clave privada en un archivo PKCs.

El comando sería openssl pkcs como parámetro export out indicándole aquí un nombre de fichero, otro como in con otro nombre de fichero, otro como inkey con otro nombre y otros dos con pass in y passout indicando unas contraseñas.

Pero vamos a explicar cada uno de estos argumentos con detalle.

Como en el comando anterior, comenzamos llamando al propio comando openssl.

Después con Pkcs indicamos que se realizará una operación relacionada con el estándar PKCS 12, que es este formato de archivos que hemos visto que iba a contener el certificado digital junto con la clave privada.

Después tendríamos el argumento export que nos especifica que se está realizando una operación de exportación hacia un archivo PKCs, esto significa que estamos tomando el certificado y la clave privada combinándolas en este formato de archivo.

Lo que estamos haciendo es ir indicar el nombre del archivo de salida donde se guardará el PKCs 12 resultante, en este caso le vamos a llamar Cert P.

Luego tendríamos el parámetro in en el cual le vamos a indicar el certificado que se utilizará para la exportación, en este caso estamos utilizando Cert Pem que es el certificado que hemos generado con anterioridad.

Tendríamos también Inkey que lo lo que estamos haciendo es especificar la clave privada correspondiente al certificado que se está exportando.

El archivo Key Pem es el que contiene la clave privada que hemos generado anteriormente.

Por último los dos parámetros que nos quedan, estos dos argumentos sería passing, que vamos a indicar la contraseña que se utiliza para desbloquear la clave privada, en este caso podríamos poner 1234.

Y por otro lado con pass lo que estamos haciendo es proporcionar la contraseña que se utilizará para proteger el nuevo archivo pkcs resultante, que por ejemplo le pasamos 1234, pero ya sabéis que tendríamos que utilizar contraseñas robustas.

En resumen, este comando toma el certificado cef pem y su clave privada key pem, los combina un único archivo pkcs que le hemos llamado cef y protegería a este archivo con la contraseña 1234, por ejemplo.

A continuación vamos a pasar a ver esto de una manera práctica.

Me encuentro en este sistema operativo kali linux y lo que vamos a hacer es poner en práctica esto que acabamos de ver.

Primero vamos a levantar un servidor normal por HTTP, vamos a ver cómo somos capaces de llegar a ver la transmisión de los datos que está viendo ese intercambio y posteriormente generaremos el nuevo certificado y vamos a tratar de volver acceder a la página y vamos a ver la diferencia.

Como podéis ver aquí ya tengo un fichero de python que nos va a levantar un servidor muy sencillo y que cuando hagamos una petición pues simplemente nos va a devolver un hello world.

Fijaos que vamos a lanzar este comando, vamos a hacer uso de python y vamos a llamar a server TIS, que es como se llama el archivo, vamos a ponerlo en ejecución y ahora nos vamos a dirigir aquí a esta página y vamos a ir a localhost, al puerto 4443.

Aquí como podemos ver se nos está respondiendo con hello world a nivel del gráfico.

¿Cómo?

¿Qué está pasando por aquí detrás?

Bueno, vamos a abrir el wireshark, este analizador de paquetes y vamos a ponernos a escuchar en la dirección de loopback, ya que nosotros estamos accediendo directamente a un servidor que estamos levantando en nuestra propia máquina y accedemos con la dirección de localhost, así que vamos a acceder por aquí, pero es que además para filtrar un poco más el tráfico le digo que quiero que me filtre por todas lo que está llegando al puerto 4443.

Ahora sí, vamos a utilizar este filtro y vamos a ponernos a capturar aquí, claro, cuando yo repito esta petición nos.

A capturar aquí, claro, cuando yo repito esta petición.

Si nos vamos al wireshark pues somos capaces de.

Vamos a parar esto.

Como veis somos capaces de ver la petición get poniendo el protocolo HTTP y seríamos capaces de poder seguir este stream y por tanto ver cuál es la información que se ha pasado.

Esto quiere decir que cualquier cosa, que si esto fuese una transacción bancaria o cualquier cosa, pues todo podríamos llegar a verlo aquí, un usuario, una contraseña o si estamos introduciendo una tarjeta de crédito, todo está pasando en lo que conocemos como texto en claro, tanto lo que mandamos como en este caso lo que recibimos.

Ya que hemos visto esto, lo que vamos a hacer ahora es lo siguiente, vamos a irnos al código, fijaos que ya tengo preparado por aquí el comando de lo que vamos a hacer para crear el certificado, así que bueno, esto no lo vamos directamente a copiar, vamos a parar este servidor, limpiamos por aquí, vamos a pegar el portapapeles y como veis estamos haciendo ahora mismo esa solicitud de certificado autofirmado RSA y vamos a tener el dinero key pen y Cert pen con una validez de 365 días.

Bueno, aquí se nos está indicando una contraseña que tenemos que poner, vamos a definirle una contraseña, ya sabéis que debería ser una contraseña robusta y es que en este caso no se ve lo que estamos tecleando.

Vamos a volver a poner la contraseña y ahora vamos a rellenar una serie de datos básicos que son necesarios para el certificado.

Si los ponemos por defecto se nos va a crear la plantilla por así decirlo, por defecto de un pero aquí podríamos indicarle por ejemplo código del país, una provincia, podemos poner por ejemplo Madrid, bueno localidad también vamos a poner Madrid, la organización la vamos a dejar por defecto aquí, la sección también, aquí tendríamos que indicar el dominio, que en este caso también le vamos a dejar por defecto, una dirección de correo, bueno tendríamos que rellenar todos los datos, aquí lo que hemos conseguido hacer es tener dos ficheros que de hecho pues las podríamos llegar a leer, ya sabemos que esto se puede leer con un editor, veis que aquí tenemos lo que sería la parte pública del

Pues si veis ya tengo esta parte del código adaptada y es que simplemente añadiéndole esta línea que vamos a ir poniendo aquí, estamos diciendo que nos coja tanto el archivo de claves como el certificado y que lo pongan o que haga uso de ssl.

Ya sabéis que ssl realmente es como se llama la librería, pero actualmente trabajamos con tls y ssl es el protocolo anterior o antiguo.

En este caso hay muchas herramientas que han permanecido con ese nombre, por ejemplo OpenSSL, no tenemos un open TLS sino se conoce como OpenSSL y así va a seguir, en este caso estas librerías que son de python son las librerías SSL y de nuevo pues así van a seguir aunque estemos trabajando con TLS.

Como veis el TLS simplemente es una evolución del SS, así que vamos a repetir, vamos a guardar, vamos a hacer aquí la limpieza y vamos a volver a lanzar el servidor de Python.

De momento le lanzamos aquí ya nos está pidiendo la contraseña que hemos puesto del certificado, que lo tenemos protegido con la contraseña y ahora sí ya estamos a la escucha y vamos a repetir aquí nos vamos a cerrar todo lo que haya pasado por aquí, lo limpiamos y vamos a iniciar de nuevo la escucha.

Así que bueno, vamos a repetir.

Veis que nos está dando un problema y es porque básicamente tendríamos que acceder a través de HTTPs.

Por aquí vemos que nos está diciendo, oye, aquí hay un problema de seguridad porque no reconocemos de quién es ese certificado.

Claro, es un certificado auto firmado, lo hemos firmado nosotros y no lo conocemos.

Sin embargo nosotros podemos ver el certificado que como veis es el que hemos creado nosotros.

Como luego no hemos llegado a escribir nada en la organización ni nada, pues bueno, pues directamente vemos como nos aparece este que está por defecto en OpensSL, sin embargo el país, la provincia y demás nos sale y por ejemplo nos salen 365 días, estamos 15 de abril de 2024 al 15 de abril del 2025.

Vamos a darle a que sí, que aceptamos el riesgo y queremos continuar de nuevo nosotros aquí vemos el Hello World, vemos el candadito como ese aviso de que hemos añadido una excepción de que confiamos en este certificado.

Esto puede ser peligroso porque si nos están haciendo una suplantación de certificado en una página legítima tendríamos un problema, eso que quede claro.

Lo que queremos simplemente demostrar aquí es que hemos añadido nuestro propio certificado en el cual nosotros confiamos y si nos vamos ahora a ver qué ha pasado en Wireshark, vemos que ya no tenemos acceso.

En este caso, bueno, esta ha sido la primera petición que hemos intentado hacer que nos ha dado el error.

Fijaos que si vamos a volver a iniciar la captura, porque la primera petición que hemos hecho ha sido por HTTP cuando debería haber ido por HTTPs, aquí sí tenemos una petición limpia, una petición que acabamos de hacer ahora y vemos como no tenemos rastro del protocolo HTTP.

Vamos a parar el wireshark, vemos como todo lo que nos aparece aquí, bueno de hecho aquí vemos ese intercambio de claves que está pasando, vemos el sync sync ack, es decir, se establece la conexión, tenemos aquí de nuevo otro intercambio de claves, pero vemos como en ningún momento vamos a tener nosotros acceso a esta información que se está transmitiendo.

De hecho, si hiciésemos aquí, por ejemplo, para intentar seguir un stream de aquí, el TCP stream, no entendemos nada de lo que está pasando.

Esto no quiere decir que por ejemplo no podamos ver otro tipo de tráfico no cifrado como podrían ser las resoluciones DNS y poder ver hacia dónde está navegando un usuario, porque al final si la petición DNS va en este caso sin cifrar, pues podríamos verlo, pero como veis, lo que es el propio contenido de lo que se está compartiendo va cifrado.

Es decir, hemos hecho ese túnel gracias al certificado y ya estamos protegiendo la comunicación porque no somos capaces de ver el tráfico que está pasando por ahí a menos que tuviésemos la clave de la sesión, que eso al final se establece entre el navegador y el servidor.

Esa clave sí que seríamos capaces de tráfico, pero alguien que esté indefectando el tráfico que está pasando entre nosotros y el servidor, esto es lo que estaría bien.

Así que como veis, es bastante sencillo el poder aplicar una seguridad básica para proteger nuestras comunicaciones, o bien si estamos en un entorno de pruebas con un certificado autofirmado o si estamos en un entorno real, en un entorno de producción, con la correspondiente petición a una autoridad certificación que nos diera un certificado válido asociado a nuestro dominio.

Para finalizar, vamos a comentar algunas conclusiones sobre esta sesión sobre los protocolos SSL y TLS y explorar cómo estos protocolos desempeñan un papel fundamental en la protección de las comunicaciones en línea.

Hemos descubierto cómo SSL y TLS establecen un canal seguro entre un cliente 1 servidor, garantizando la autenticidad, confidencialidad e integridad de los datos transmitidos.

En su funcionamiento.

Se realiza un proceso de saludo y se procede a iniciar la transferencia segura de los datos cifrados.

Por otro lado, se ha explorado la generación de certificados autofirmados con la herramienta OpenSSL y se ha comprendido cómo esta herramienta ofrece una solución segura y eficiente para entornos de desarrollo y de pruebas.

La capacidad de generar certificados rápidamente y sin depender de una autoridad de certificación externa nos proporciona una mayor independencia y control sobre la gestión de la seguridad de nuestros sistemas.

Otra cosa a tener en cuenta son los formatos de los certificados.

Al considerar las ventajas de convertir un certificado en formato PEM a otro en formato P, se ha reconocido la importancia de la portabilidad y la facilidad de gestión que esta conversión ofrece.

Al combinar el certificado y la clave privada en un solo archivo protegido por contraseña, podemos garantizar una mayor eficiencia en la gestión de nuestros certificados digitales.

A medida que avanzamos en este panorama digital que está en constante evolución, la seguridad en línea seguirá siendo una prioridad crítica.

Al comprender y aplicar los principios de SSL o TLS, estamos mejor equipados para asegurar nuestras comunicaciones en Internet y proteger nuestra privacidad y seguridad en el mundo digital de hoy.

Y con esto llegamos al final de la sesión.

Os esperamos en el siguiente vídeo.