{desafío} latam_

SSL

Objetivos

- Conocer las implicaciones de SSL.
- Conocer la importancia de **no** enviar información sensible sin SSL.
- Entender la importancia de verificar el certificado.

Introducción

SSL es un acrónimo de Secure Sockets Layer, es una capa de seguridad que establece cifrado (encriptación) entre el cliente y el servidor y evita que la información que enviamos o recibimos sea leída por un tercero.

Esto sucede ya sea conectándonos a una página web por el navegador o una API.

Es importante que toda comunicación que maneje información sensible ocupe esta capa de cifrado debido a que es muy fácil incluso para inexpertos interceptar estas comunicaciones.

Vamos a revisar en términos generales como funciona el sistema, pero esto no es necesario para aprovechar las bondades de la seguridad.

Cómo funciona la encriptación por SSL

La clave para entender el funcionamiento de la encriptación es entender que hay dos claves, una para cifrar y otra descifrar. Para entender esto veámoslo con un ejemplo.

Supongamos por un momento que la comunicación es entre dos persona, Alice y Bob. Para cifrar el mensaje Alice tiene una clave y para descifrarlo tiene otra. Previo a comunicarse Alice se junta con Bob y le traspasa la clave para descifrar.

Esto permite que Alice cifre su mensaje con su clave, le envía el mensaje a bob y bob la descifra con la clave que la pasó previamente Alice. Esto tiene dos ventajas, primero el mensaje pasa cifrado por lo que nadie mas puede leerlo a menos que tenga la clave para descifrar, pero además tiene otra ventaja, la llave para descifrar solo sirve para descifrar mensajes de Alice, por lo que sabremos que el mensaje viene realmente de Alice y no de otra persona.

Clave pública, clave privada

Las dos claves son distintas por eso decimos que este sistema de cifrado es asimétrico. Una clave es para cifrar el mensaje la otra para descifrarlo.

Las claves tienen nombre, la clave que firma el mensaje pero jamás se comparte recibe el nombre de clave privada y la clave que se comparte recibe el nombre de clave pública.

Ventajas de SSL

El sistema de juego entonces tiene dos ventajas.

- Cifra el mensaje impidiendo que terceros puedan leerlo.
- Asegura que el emisor es quien dice ser, porque si alguien mas cifró el mensaje con una llave distinta, el mensaje no tendrá sentido al decifrarlo con la llave pública.

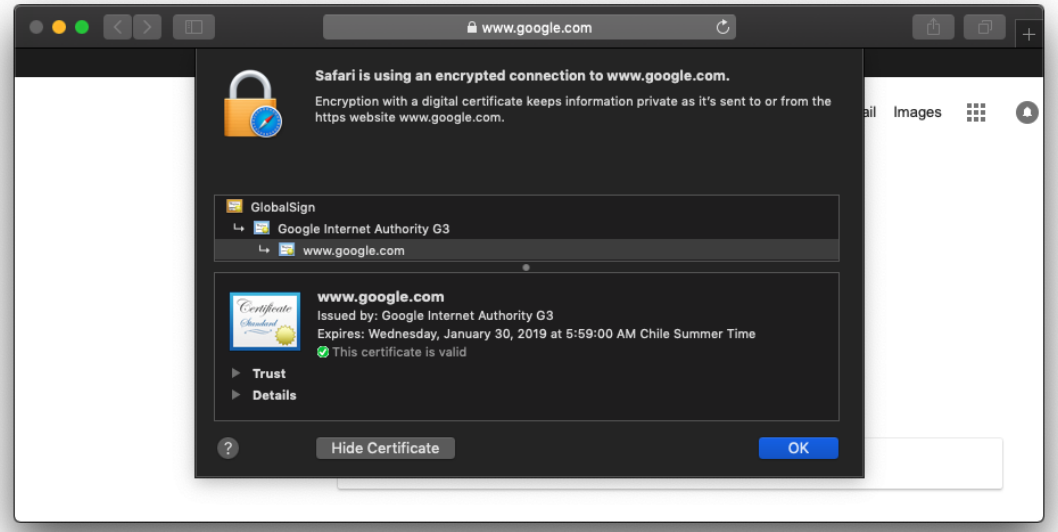
Todo el sistema funciona de forma automática

Todo lo que hemos hablado hasta ahora funciona de forma automática. Y al realizarlo vía código es casi automático, tendremos que agregar solo 2 líneas extra de código que explicitan que utilizaremos SSL.

La magia que sucede detrás

Al momento de conectarnos a un sitio web que utilice SSL con un navegador se establece un acuerdo, llamado en inglés **handshake** de forma automática. En este handshake el servidor envía un certificado que tiene el nombre del sitio y la clave pública al cliente.

Si nos conectamos con el navegador y hacemos click en el candado de la conexión segura podemos ver el certificado.



El cliente ocupa la clave pública que viene en el certificado para cifrar un mensaje y se la devuelve al servidor, si el servidor puede descifrar el nuevo mensaje significa que el servidor tiene la clave privada correcta.

Dentro del proceso ambos, cliente y servidor utilizando los números que se enviaron durante el handshakre crearon un clave secreta especial, el resto del proceso ocupa una sistema de cifrado simétrico utilizando esta única clave que nunca fue transmitida pero que ambos servidores tienen.

Conectándose con SSL

Postman identifica automáticamente si un request ocupa https y genera el código para conectarnos.

Lo vamos a copiar y dejar dentro del método

```
def request(url_requested)
  url = URI(url_requested)

  http = Net::HTTP.new(url.host, url.port)
  http.use_ssl = true # Se agrega esta línea
  http.verify_mode = OpenSSL::SSL::VERIFY_NONE # Se agrega esta otra línea

  request = Net::HTTP::Get.new(url)
  request["cache-control"] = 'no-cache'
  request["postman-token"] = '5f4b1b36-5bcd-4c49-f578-75a752af8fd5'
```

```
  response = http.request(request)
  return JSON.parse(response.body)
end
```

Veremos que la única diferencia con nuestro requests previos son estas líneas

```
http.use_ssl = true # Se agrega esta línea
http.verify_mode = OpenSSL::SSL::VERIFY_NONE # Se agrega esta otra línea
```

Esto es lo único que necesitamos para establecer una conexión segura, bueno, casi, todavía queda resolver el tema de la validez del certificado.

Hombre en el medio

Hombre en el medio (**Man in the middle**) es una estrategia de hacking que consiste en hacer de puente (**Proxy**) entre el cliente y el servidor.

El atacante se sitúa en el punto medio, captura la clave pública que envía el servidor y le envía una nueva clave pública al cliente. El cliente cree que se está conectando con el servidor directamente pero no es así y toda la comunicación queda comprometida.

Este ataque es muy fácil de lograr pero solo si el certificado no es validado (o auto firmado). Al momento de conectarse los certificados se revisan contra agentes certificadores. Un certificado validado (nombre de sitio + clave pública) complica la situación al atacante puesto que este difícilmente tendrá la clave privada asociada al certificado.

Protegiéndose del Hombre en el medio

El ataque de **Man in the middle** es fácil de lograr si el certificado no se verifica, por lo que esta línea es peligrosa.

```
http.verify_mode = OpenSSL::SSL::VERIFY_NONE
```

Es útil si estamos probando una API que estemos desarrollando, pero si estamos enviando información sensible (como por ejemplo un login, un password o un token) entonces es peligroso.

Podemos cambiar la línea por

```
http.verify_mode = OpenSSL::SSL::VERIFY_PEER
```

Corrigiendo nuestro código, finalmente nuestro método quedaría así:

```
require 'uri'
require 'net/http'

def request(url_requested)
  url = URI(url_requested)

  http = Net::HTTP.new(url.host, url.port)
  http.use_ssl = true
  http.verify_mode = OpenSSL::SSL::VERIFY_PEER

  request = Net::HTTP::Get.new(url)
  request["cache-control"] = 'no-cache'
  request["postman-token"] = '5f4b1b36-5bcd-4c49-f578-75a752af8fd5'
  response = http.request(request)
  return JSON.parse(response.body)
end
```

```
data = request('https://jsonplaceholder.typicode.com/photos')[0..5] # Tomemos solo 5
```

Veremos como resultado los primeros 5 elementos


```
[{"albumId"=>1,
  "id"=>1,
  "title"=>"accusamus beatae ad facilis cum similique qui sunt",
  "url"=>"https://via.placeholder.com/600/92c952",
  "thumbnailUrl"=>"https://via.placeholder.com/150/92c952"},
    ...
```