Security test

<https://www.soapui.org/security-testing/getting-started.html>

Morpheus security

<http://www.drchaos.com/morpheus-man-in-the-middle-security-tool/>

--- ventajas ssl MITM

<https://www.blackmoreops.com/2015/12/22/man-in-the-middle-attack-using-kali-linux/>

MONGO DB AND SPRING BOOT

<https://www.petrikainulainen.net/programming/spring-framework/creating-a-rest-api-with-spring-boot-and-mongodb/>

man in the middle attack

<https://www.youtube.com/watch?v=XW5naPY4FsA>

Over the years a bunch of students don't really 'care about encrypting the passwords' or avoid using SSL as a possible solution (i.e., an SHA-128/256 hash of the password is sent over the wire in case of plain text but still no SSL. So I can conduct a replay attack as a result).

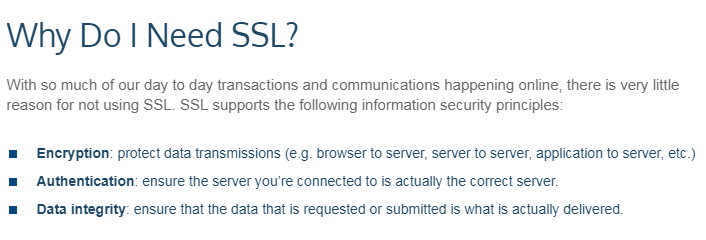
I want to be able to 'demo the dangers' of the man-in-the-middle (MITM) and am not sure how to go about it. Students question if someone is really able to pull it off and are curious on the *how*. I thought it'd be good to 'show it to them'.

You can also use SSLstrip which will attempt to remove the SSL protection on some websites and give you access to clear text credentials.

<https://security.stackexchange.com/questions/31653/how-to-demonstrate-man-in-the-middle-attack>

SSL is better

--------------------------------------



Benefits of encript

<http://newsroom.kaspersky.eu/fr-be/news/news-article/article/encryption-five-major-benefits-2/>

**Benefit 4: Guarantee Data Integrity**

Targeted data theft is one thing, but another way to misuse data is through manipulation. Even though a hacker may have absolutely no interest in the information in question, he or she can manipulate specific data to disrupt corporate communications. If encrypted data is used, the recipient will definitely notice that it has been tampered with.

**https protege de man in the middle**

Https is based on public/private-key cryptography. This basically means that there is a key pair: The public key is used for encryption and the secret private key is required for decryption

ssl in web service

https://stackoverflow.com/questions/6810634/how-to-make-simple-ssl-thru-web-services

//---------------------------------------------

foro

<https://stackoverflow.com/questions/16690442/encrypting-exchanged-data-in-a-restful>

# [Encrypting exchanged data in a RESTful](https://stackoverflow.com/questions/16690442/encrypting-exchanged-data-in-a-restful)

I have created a Restfull web-service in java that is consumed by an android application. The work is almost done.

I recommend you to use **HTTPS** for your need. **SSL/TLS provide better encryption decryption over https**. **And you don't need to reinvent wheels for this basic purpose**. If you are using **tomcat to host REST web serivces then here is a simple tutorial to turn on SSL in tocamt**

**Make tomcat SSL**

<https://dzone.com/articles/setting-ssl-tomcat-5-minutes>

//-------------------

<https://security.stackexchange.com/questions/38388/ssl-vs-encryption>

I am working on creating an android mobile application which needs to connect to a server from time to time. The application makes use of sensitive data which I would not like to get into the wrong hands.

Using [Android's DefaultHttpClient](https://developer.android.com/reference/org/apache/http/impl/client/DefaultHttpClient.html) you can establish an HTTPS connection and configure it to trust only certain certificates. It's quite easy to use and it's already there. You can even use [SSLSocket](https://developer.android.com/reference/javax/net/ssl/SSLSocket.html) if you don't fancy HTTP. The main point is: **don't do your own crypto**.

As for the overhead, don't worry, most of the time it's negligible. Your data is sensitive and the communication will happen from time to time. As far as I can see, there's nothing against using SSL here.

**La mayoría de tiempo es insignificante.**

The best solution for transport security is TLS/SSL. It's supported by default and the overhead is minimal. **It support implicit authentication as well.**

**HTTPS PERFORMANCE**

**https://www.keycdn.com/blog/https-performance-overhead/**

<https://stackoverflow.com/questions/149274/http-vs-https-performance>

Measure (using a tool such as Firebug) the page load times while the server is on the end of a simulated high-latency link. Tools exist to simulate a high latency link - for Linux there is "netem". Compare HTTP with HTTPS on the same setup.

The latency can be mitigated to some extent by:

* Ensuring that your server is using HTTP keepalives - this allows the client to reuse SSL sessions, which avoids the need for another handshake
* Reducing the number of requests to as few as possible - by combining resources where possible (e.g. .js include files, CSS) and encouraging client-side caching
* Reduce the number of page loads, e.g. by loading data not required into the page (perhaps in a hidden HTML element) and then showing it using client-script.

<https://www.tunetheweb.com/blog/http-versus-https-versus-http2/>

Se entiende que el impacto en el rendimiento de HTTPS ya no es una barrera para la adopción de sitios web.

http2 improve performance in https tan http

--Prueba de rendimiento HTTPS

<http://www.httpvshttps.com/>

Por lo que realmente se parece HTTPS causar ningún impacto notable de rendimiento para sitios web sencillos, como ha sido declarado por la mayoría de los expertos desde hace algún tiempo.

El otro punto obvio es que HTTP / 2 es **mucho** más rápido que HTTP o HTTPS, aunque sólo está disponible en HTTPS para los navegadores web. Esto es lo que sentí fue el punto engañoso de la [https://www.httpvshttps.com/](https://translate.googleusercontent.com/translate_c?act=url&depth=1&hl=es&ie=UTF8&prev=_t&rurl=translate.google.com&sl=en&sp=nmt4&tl=es&u=https://www.httpvshttps.com/&usg=ALkJrhhYhK7evYvOgRwhijgZmH63__72bA) sitio web, ya que no fue HTTPS que causó el desempeño presumir, pero es realmente impresionante lo mucho que hace una diferencia HTTP / 2 - Incluso en un sitio más pequeño de 36 imágenes. Por supuesto, el sitio de 360 ​​imágenes sigue siendo un ejemplo extremo y no vas a ver un 91% de mejora simplemente encendiendo HTTPS - incluso con HTTP / 2 para la mayoría de los sitios. Sin embargo incluso el sitio más pequeño de la imagen 36 ve una mejora enorme del 70% en HTTP / 2 - mucho más que habría esperado.

HTTPS en particular es una tecnología bien establecida ahora y con iniciativas como [Let's Encrypt](https://translate.googleusercontent.com/translate_c?act=url&depth=1&hl=es&ie=UTF8&prev=_t&rurl=translate.google.com&sl=en&sp=nmt4&tl=es&u=https://letsencrypt.org/&usg=ALkJrhiQxhvogV2vfvGtGDiuSi1aGlToCA) nunca ha sido más fácil o más barato obtener un certificado HTTPS para su sitio web. Hay [muchas razones para usar HTTPS](https://translate.googleusercontent.com/translate_c?act=url&depth=1&hl=es&ie=UTF8&prev=_t&rurl=translate.google.com&sl=en&sp=nmt4&tl=es&u=https://scotthelme.co.uk/still-think-you-dont-need-https/&usg=ALkJrhg46uHzN9_GLUeubPMv0bdKAF_zTQ) y va a ser más difícil y más difícil mantener un sitio HTTP solo como[funciones de reserva de](https://translate.googleusercontent.com/translate_c?act=url&depth=1&hl=es&ie=UTF8&prev=_t&rurl=translate.google.com&sl=en&sp=nmt4&tl=es&u=https://developers.google.com/web/updates/2016/04/geolocation-on-secure-contexts-only%3Fhl%3Den&usg=ALkJrhiJ64hhPwXWRuWIhowo8aU77kpZwA) navegadores web [para sitios HTTPS](https://translate.googleusercontent.com/translate_c?act=url&depth=1&hl=es&ie=UTF8&prev=_t&rurl=translate.google.com&sl=en&sp=nmt4&tl=es&u=https://developers.google.com/web/updates/2016/04/geolocation-on-secure-contexts-only%3Fhl%3Den&usg=ALkJrhiJ64hhPwXWRuWIhowo8aU77kpZwA) . Si usted no está en HTTPS entonces usted debe mirarlo ahora. Estas pruebas demuestran que el impacto en el rendimiento no es ni siquiera notable para la mayoría de los sitios

API REST Y HTTP/2

<https://www.api2cart.com/blog/rest-api-services-can-expect-http2/>

varias

<https://stackoverflow.com/questions/31692868/rest-api-with-http-2>

<https://community.atlassian.com/t5/JIRA-Core-questions/Rest-API-not-working-with-HTTP-2/qaq-p/83822>

<https://www.smashingmagazine.com/2016/02/getting-ready-for-http2/>

<https://www.quora.com/How-does-HTTP-2-affect-RESTful-APIs>