HTTPS Everywhere

In this lesson, we'll look at the importance of HTTPS more closely.

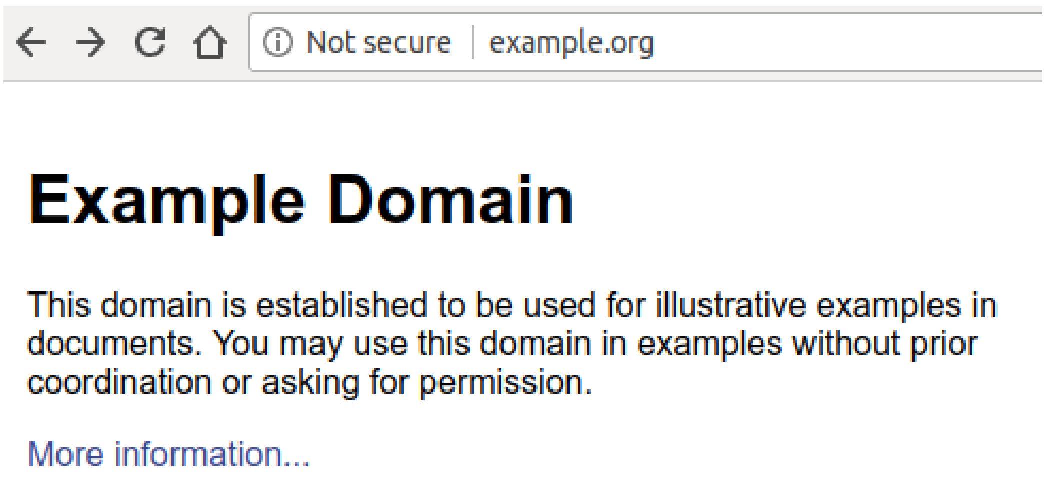
We'll cover the following

- Discontinued acceptance of HTTP

Discontinued acceptance of HTTP

Still debating whether you should support HTTPS on your website? I don't have good news for you; browsers have started to push users away from websites that don't support HTTPS in order to force web developers towards providing a fully encrypted browsing experience.

Behind the motto "*HTTPS everywhere*", browsers started to take a stand against unencrypted connections. Google was the first browser vendor who gave web developers a deadline, announcing that starting with Chrome 68 (July 2018) it would mark HTTP websites as "not secure".



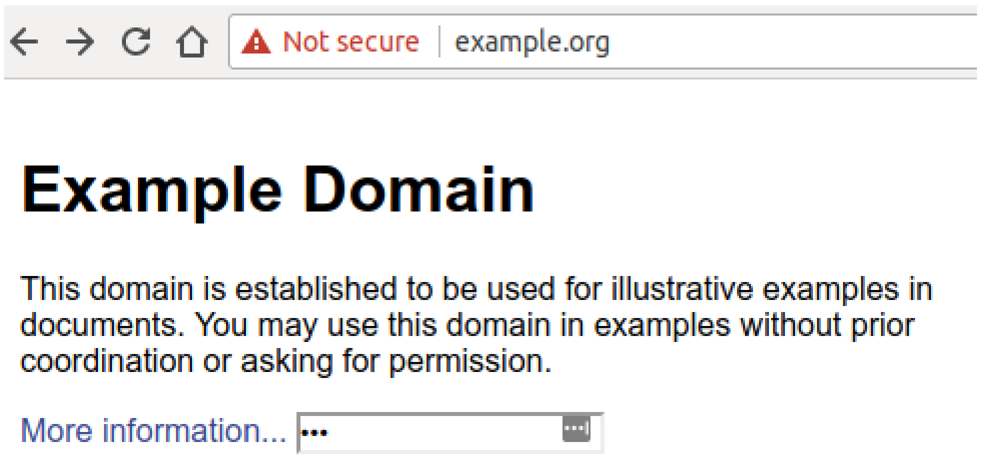
Example Domain

This domain is established to be used for illustrative examples in documents. You may use this domain in examples without prior coordination or asking for permission.

More information...

Chrome's warning when browsing a website via HTTP

Even more worrying for websites not taking advantage of HTTPS is the fact that, as soon as the user inputs anything on the webpage, the "Not secure" label turns red, a move that should encourage users to think twice before exchanging data with websites that don't support HTTPS.


Not secure | example.org

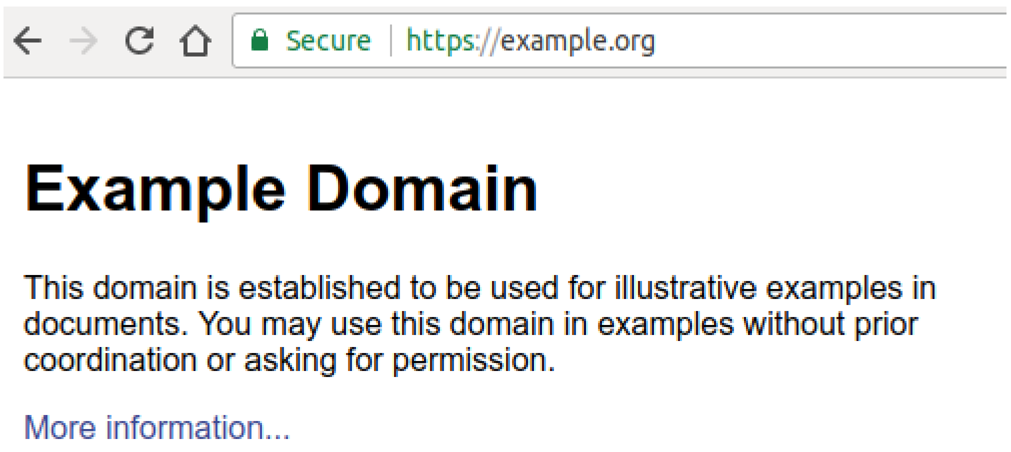
Example Domain

This domain is established to be used for illustrative examples in documents. You may use this domain in examples without prior coordination or asking for permission.

More information...

When typing on an input field in a non-HTTPS website, Chrome will mark the "not secure" warning in red

Compare this to what a website equipped with a valid certificate and running on HTTPS looks like.


Secure | https://example.org

Example Domain

This domain is established to be used for illustrative examples in documents. You may use this domain in examples without prior coordination or asking for permission.

More information...

The green lock and the "Secure" label invite users to trust a website

In theory, a website does not have to be secure; in practice, this scares users away. Rightfully so! Back in the day, when H2 was not a reality, it could have made sense to stick to unencrypted, plain HTTP traffic. Nowadays, the downsides of doing so are very dangerous, as unwanted ads or crypto-mining software can be injected by attackers or even ISPs when users browse unencrypted websites. That is impossible to do when a website uses HTTPS. Join the *HTTPS everywhere* movement and help us make the web a safer place for surfers. I recommend you read "Here's why your static website needs HTTPS" by Troy Hunt, a post that highlights the dangers of not using encrypted connections when communicating with any website.

🔑 Use HTTPS

Securing our customers' experience is a high priority when developing web apps. By using HTTPS, you ensure that the information the user exchanges with your application is transferred securely over the network.

ℹ CloudFlare: understanding what security really means

You might have heard of CloudFlare's free SSL certificate offering, as it encouraged hundreds of thousands of webmasters to serve their websites through HTTPS. Similar to their free DDoS protection service, CloudFlare's service was a breakthrough for small site owners as SSL certificates used to be quite expensive and many webmasters decided it wasn't worth offering HTTPS to their customers.

You might think CloudFlare makes the internet a safe place, but I argue that even though it improves the overall security of sites, problems can still happen. Webmasters might be oblivious to them due to the idea that their site is served through HTTPS.

In a traditional network setup, the connection between the customer and your servers is secured until you reach your own network. Your load balancer typically does the SSL termination and then forwards the plain, unsecured request to one of your servers. This was an acceptable tradeoff as you would not expose those servers publicly. Intercepting the unencrypted messages would mean breaking into your secured network, a fairly hard task.

With CloudFlare, things change; the connection is secured between the client and CloudFlare's edge server, but then flies unencrypted over the internet towards your HTTP-only server. This means that if someone can intercept traffic between you and CloudFlare, they can sniff unencrypted traffic.

Does this mean CloudFlare offers a bad service? Not at all, it simply means we need to understand it before thinking that using it offers the greatest level of protection out there.

Scott Helme perfectly summarized the issue in a blog post titled "My TLS conundrum and why I decided to leave CloudFlare", which I'd recommend you read.

In the next lesson, we’ll compare and contrast GET and POST requests.