Deprecating Non-Secure HTTP

Frequently Asked Questions

Since we announced our plan to focus on HTTPS going forward, there have been some questions about the state of HTTPS in the web today, and the implications of this plan for web developers. This FAQ attempts to answer the most common questions.

Q. Does this mean my unencrypted site will stop working?

Not for a long time. Transitioning the web to HTTPS is going to take some time. The first thing we're going to do is require HTTPS for new features. So whatever your website does today, it will still work for months or years.

In the long run, there is some discussion of removing or limiting features that are currently available to unencrypted sites. Those changes will be announced well ahead of any change, so you'll have time to update your site either to not rely on those features or, we hope, to move to HTTPS. And any such changes will be made only after consultation with the web community, to make sure we're striking an appropriate balance between functionality and security.

Q. Why are you forcing me to buy a certificate? Isn't this hard on small sites?

If you want to use HTTPS, you'll have to get a certificate. That doesn't mean you have to buy one though! There are multiple free certificate providers in the market right now (e.g., StartSSL, WoSign, and soon Let's Encrypt). Some web platforms will provide you a certificate for free (e.g., Cloudflare). For those who prefer to run their own server, Mozilla already offers an HTTPS configuration generator.

Q. Won't HTTPS make my site slower?

HTTPS is basically HTTP plus encryption, so the cost of doing HTTPS is non-zero. However, on modern platforms, it is very, very small.

For many websites, encryption will actually be a gateway to **better** performance. HTTP/2 offers significant performance improvements over HTTP/1.1, and in all current browsers it is only available for encrypted sites.

Q. Why are you bothering with pushing HTTPS, given that the CA system is so broken?

For all its flaws, the CA system we have is one that underpins the entire online economy we have today. It's broken in ways, but for the most part, it works.

We are always trying to improve the quality of the Mozilla CA Program, from kicking out misbehaving CAs to strengthening disclosure requirements. And we're interested in better technologies (such as DANE and CT) for improving authentication in the long run. Even as these improvement efforts continue, the PKI as it exists today is strong enough to move forward with promoting HTTPS.

Q. Isn't this making life harder for small websites and reducing free speech?

As noted above, the general trend in the industry is that HTTPS is getting easier to deploy. Even for legacy content, there's HSTS and upgrade-insecure-requests to make the migration smoother.

As far as free speech, it's noteworthy that much of the research in anti-censorship recently has been focused on making communications more private (see, for example, SecureDrop). So the increased use of encryption seems like a good thing for free speech.

Q. What about development/corporate environments?

You'll be able to configure the browser to work for these cases. The notion of "secure" enforced by the browser in this case will be the one defined by the W3C's Privileged Contexts specification, which we expect will have a provision for local policy -- that is, for the user to configure a certain context as explicitly trusted. Combine that with the existing mechanisms for adding trusted roots, and it should be straightforward for a developer or IT guy to set up a secure environment, like Mozilla does.

Q. But there's nothing secret on my site! Why should I bother with encryption?

HTTPS isn't just about encryption. It also provides integrity, so your site can't be modified, and authentication, so users know they're connecting to you and not some attacker. Lacking any one of these three properties can cause problems. Better use of security would prevent:

- Comcast injecting ads into their customers' web traffic;
- AT&T tracking their users' browsing habits; and
- The "Great Cannon of China" knocking Github offline.

In other words, as long as your site is not secure, it can be used as a weapon against your users and against other web sites. More non-secure sites means more risk for the overall Web.

Q. If you like security so much, why are you so hard on self-signed certificates?

Self-signed certificates aren't inherently bad. If you go to the effort of manually checking that it's the right certificate, it can be more secure than a CA-issued certificate.

So why does the browser present such a scary warning? The problem is that browser doesn't know when it's supposed to be getting a self-signed certificate, and when it's supposed to be getting a CA-issued certificate. In practice, only a few legitimate sites present self-signed certificates, since manual checking is hard. That means the browser has to assume, by default, that it should be getting a CA-issued certificate, and that the self-signed certificate is suspect. Likewise, most users can't be expected to look at the details of a certificate and evaluate whether it's the right one or not, so the browser has to try to make a good decision on the user's behalf.

Self-signed certificates are fine if you tell the browser about them. Either configure them in advance (Preferences > Advanced > Certificates > Servers), or click through the scary dialogue. Of course, if you have ideas for how to improve the user experience here, please let us know.

Q. What about my home router? Or my printer?

The challenge here is not that these machines can't do HTTPS, it's that they're not provisioned with a certificate. A lot of times, this is because the device doesn't have a globally unique name, so it can't be issued a certificate in the same way that a web site can. There is a legitimate need for better technology in this space, and we're talking to some device vendors about how to improve the situation.

It should also be noted, though, that the gradual nature of our plan means that we have some time to work on this. As noted above, everything that works today will continue to work for a while, so we have some time to solve this problem.

Q. Isn't HTTPS just, like, totally the bee's knees?

Yes. Yes it is.