Web Protocols: HTTP and HTTPS

The article was created for novice system administrators who are new to computer networking and who may find it difficult to read the official specifications. The article provides information on **HTTP** protocol, its extension **HTTPS**, and instructions for **switching from HTTP to HTTPS**.

Table of Contents

[An overview of HTTP 1](#_Toc63156759)

[An Example of Interception 1](#_Toc63156760)

[An overview of HTTPS 2](#_Toc63156761)

[Benefits of HTTPS 2](#_Toc63156762)

[Switching from HTTP to HTTPS 3](#_Toc63156763)

[Configuring HTTPS for a Tilda website 3](#_Toc63156764)

[Before installation of an SSL certificate: 3](#_Toc63156765)

[If you purchased a domain on Tilda: 3](#_Toc63156766)

[If you purchased a domain by another provider: 5](#_Toc63156767)

[Configuring HTTPS for a Nginx website 7](#_Toc63156768)

[Summarized Information 9](#_Toc63156769)

# An overview of HTTP

HTTP is a data transfer protocol used for transferring any kind of data on the Internet. The abbreviation stands for “hypertext transfer protocol”, as originally HTTP was created for the transmission of hypertext documents. The standard port for HTTP connections is TCP port 80.

The protocol uses a client-server type of data transfer structure. A user initiates the connection and sends a request to the server, which processes the data, generates a response, and delivers it back to the user. HTTP is an open data transfer protocol, where requests and responses are sent in plaintext. That means the information accessible for interception, and anyone monitoring the session can read or distort the transmitted data.

# An Example of Interception

A common example of interception is a man-in-the-middle (MitM) attack. This is a type of attack in computer security, in which an attacker secretly connects to the channel between the client and the host to monitor or interfere with the data transfer protocol by deleting or distorting the information. This is particularly a threat to mail services and commercial sites selling goods or services as users enter their personal information. In this case using the HTTPS extension is preferable.

# An overview of HTTPS

HTTPS is not a separate protocol but an extension of HTTP. The standard port for HTTPS is 443 port. The extra “S” in the abbreviation stands for “Secure”, as HTTPS provides security and privacy when exchanging information between the user and the host. This is guaranteed by implementation of cryptographic protocols, such as “secure sockets layer” (SSL) or the upgraded version “transport layer security” (TLS), on top of the data transfer protocol.

Cryptographic protocols encrypt the transmitted data what protects it against attacks based on sniffing the network connection. Encryption occurs both ways that means data is always in encrypted form in between delivery. The decryption requires a session key that is generated in the process of establishing a connection between the client and the server.

# Benefits of HTTPS

HTTPS must be on every new website, as this is the standard. Besides HTTPS secures web traffic and allows the transmission of personal information, it has other advantages over HTTP.

Modern web browsers mark all HTTP websites as “not secure” and give users a warning, which may scare some of them away (see Image 1). If a site is protected by an SSL certificate, the browser displays an icon with a closed padlock in the address bar. Web browsers also limit important features for unprotected websites such as the use of geolocation. Besides, search engines consider using SSL as a ranking factor displaying HTTPS websites higher in search results.

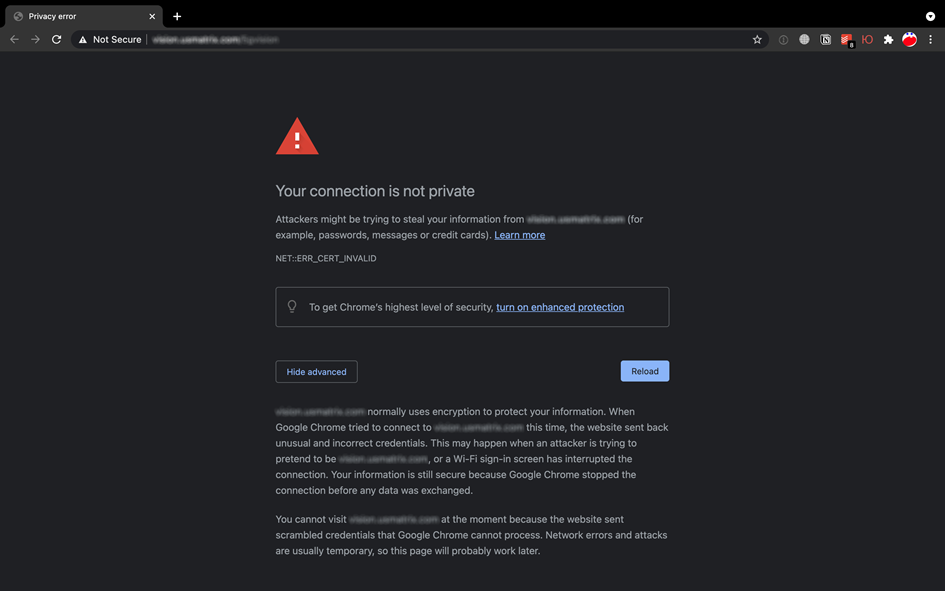


Image 1: Chrome warns of an insecure website

# Switching from HTTP to HTTPS

This section provides two step-by-step instructions for switching from HTTP to HTTPS. The first one is for websites created using a site builder called Tilda , and the second is for websites hosted on your own server running Nginx.

## Configuring HTTPS for a Tilda website

### Before installation of an SSL certificate:

1. Go to **Site Settings** → **SEO** → Setup redirects with code 301 and restore the "default" value.
2. Go to **Site Settings** → **SEO** → Make sure that at the bottom both **HTTPS** and **WWW** schemes are unchecked for robots.txt.

### If you purchased a domain on Tilda:

You already have an SSL certificate by default.

1. Go to **Domains** → **Domain details and settings** → **Use IP HTTPS**. (see Images 1,2,3)

The website name will be updated within 24 hours.

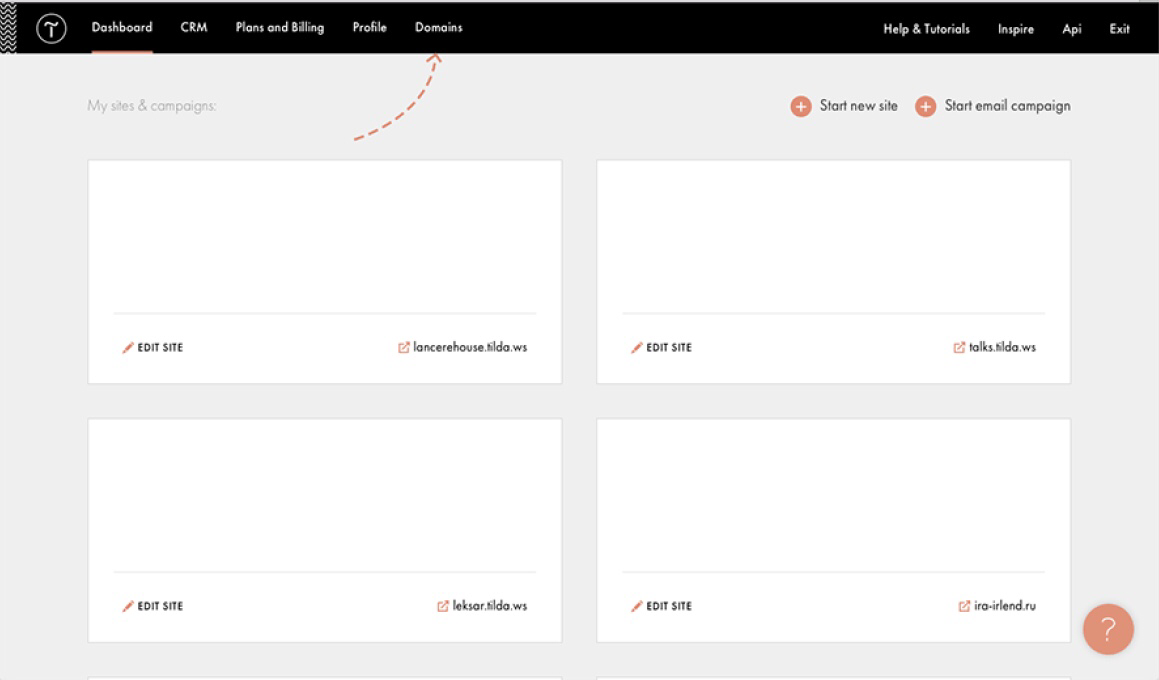


Image 2: Go to Domains

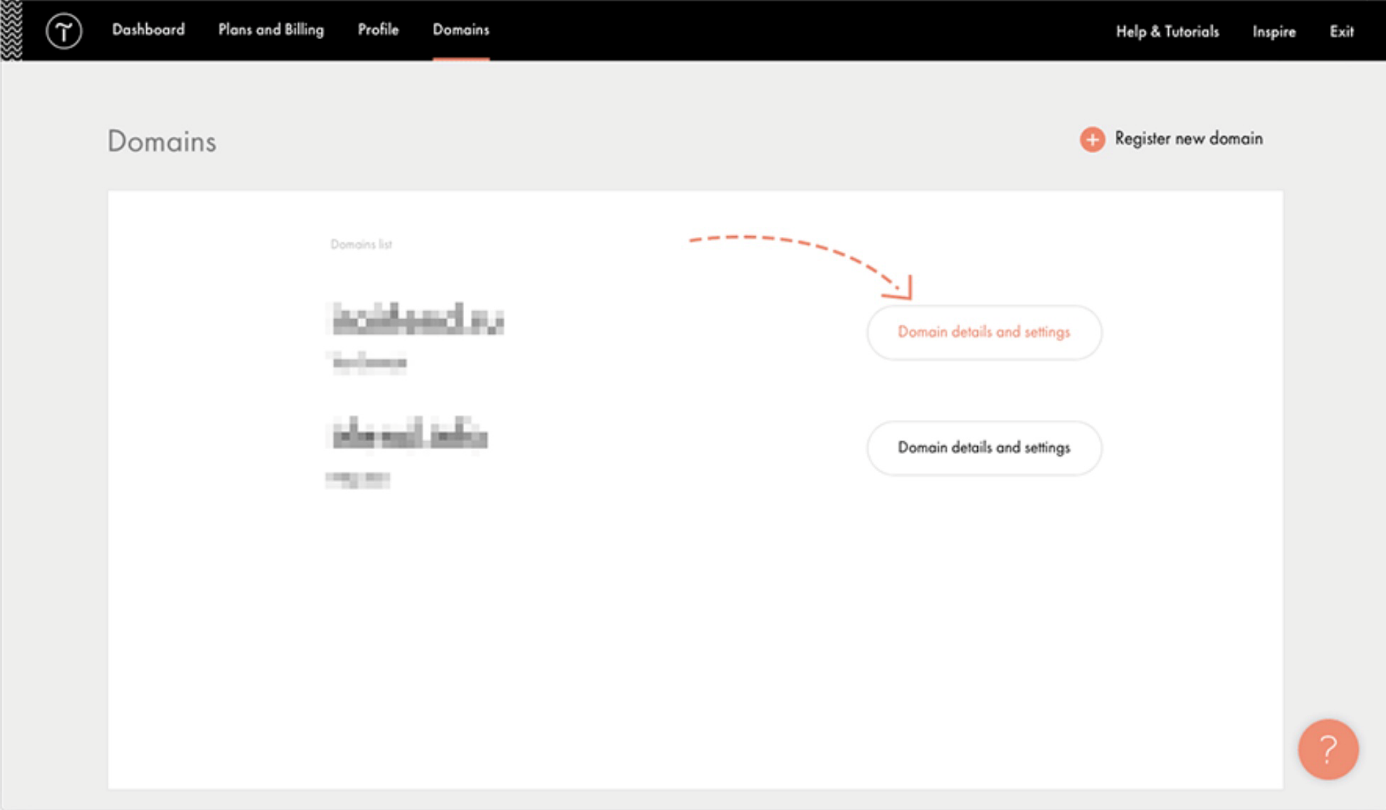


Image 3: Click on Domain details and settings

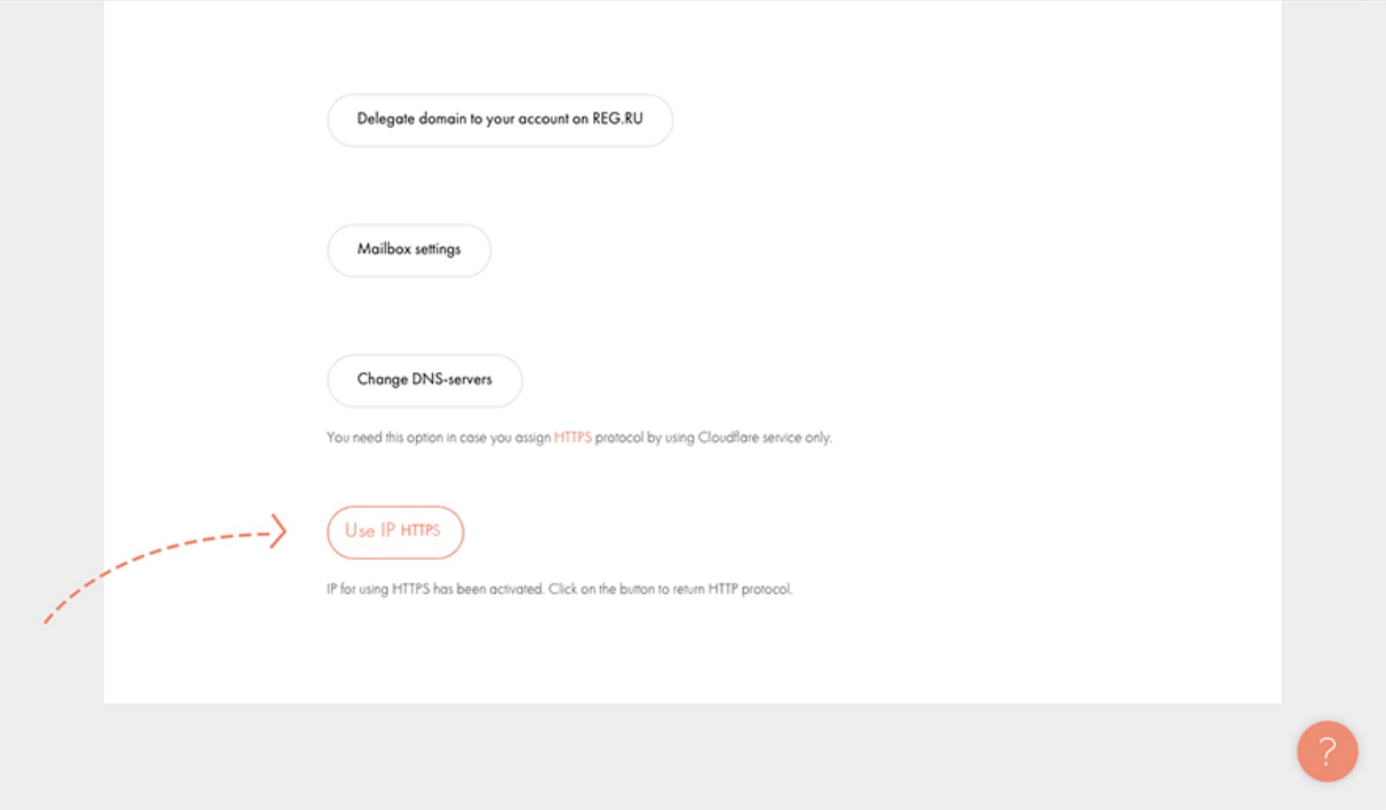


Image 4: Click on Use IP HTTPS

### If you purchased a domain by another provider:

1. Go to the website of your domain registrar and open **DNS** or **Zone Management** in your personal account.
2. Change the IP address for an **A record** (see Image 5) to:

185.203.72.17 — for the European Union

185.165.123.36 — for international users, including Russia

77.220.207.191 — for Kazakhstan

1. Change the IP address for an **A record for www** **subdomain** to the same IP address as in step 2.

The website name will be updated within 24 hours.

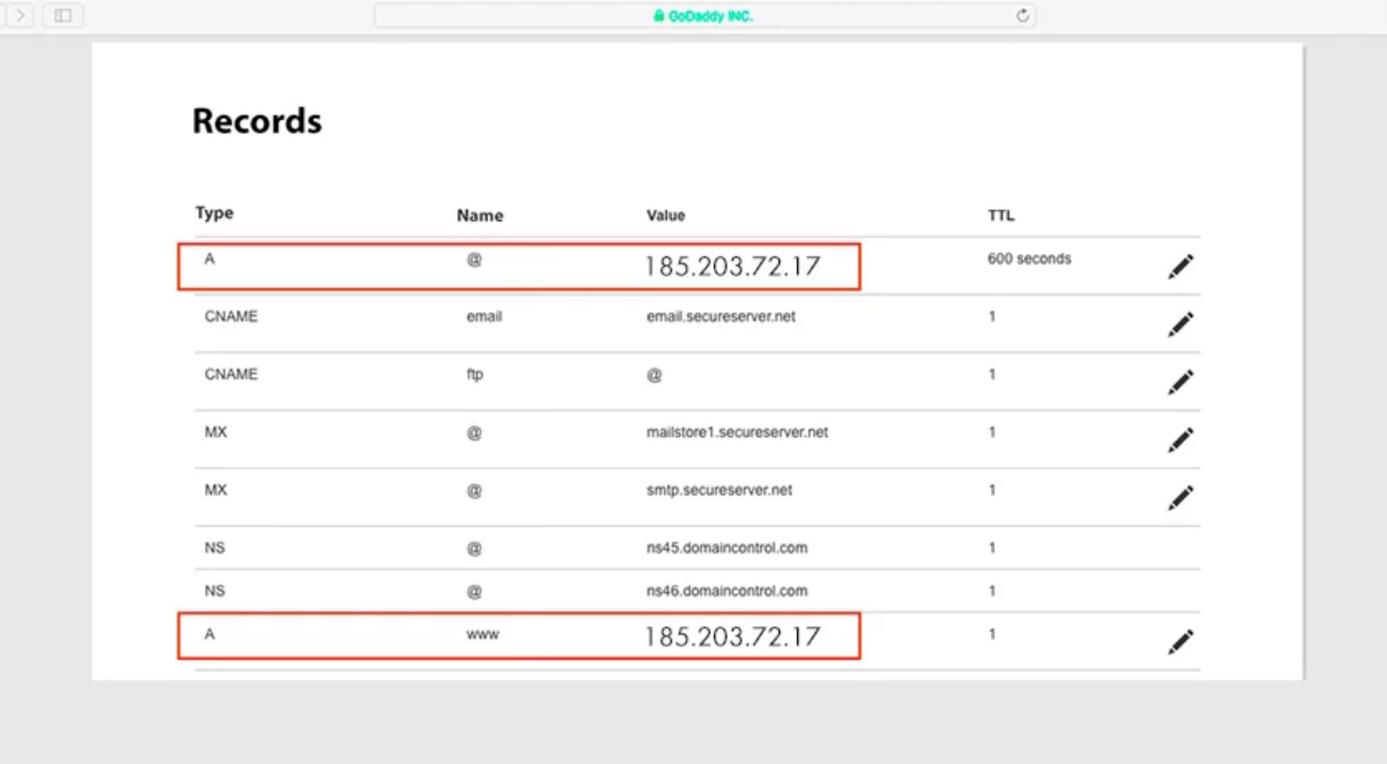


Image 5: Change the IP address for an A record and for an A record for www subdomain

1. After 24 hours, go to **Site Settings** → **SEO** →**HTTPS settings** on Tilda.
2. Click **Manage** and **enable HTTPS** for the website (see Images 6, 7)

The certificate will be connected within 30 minutes.

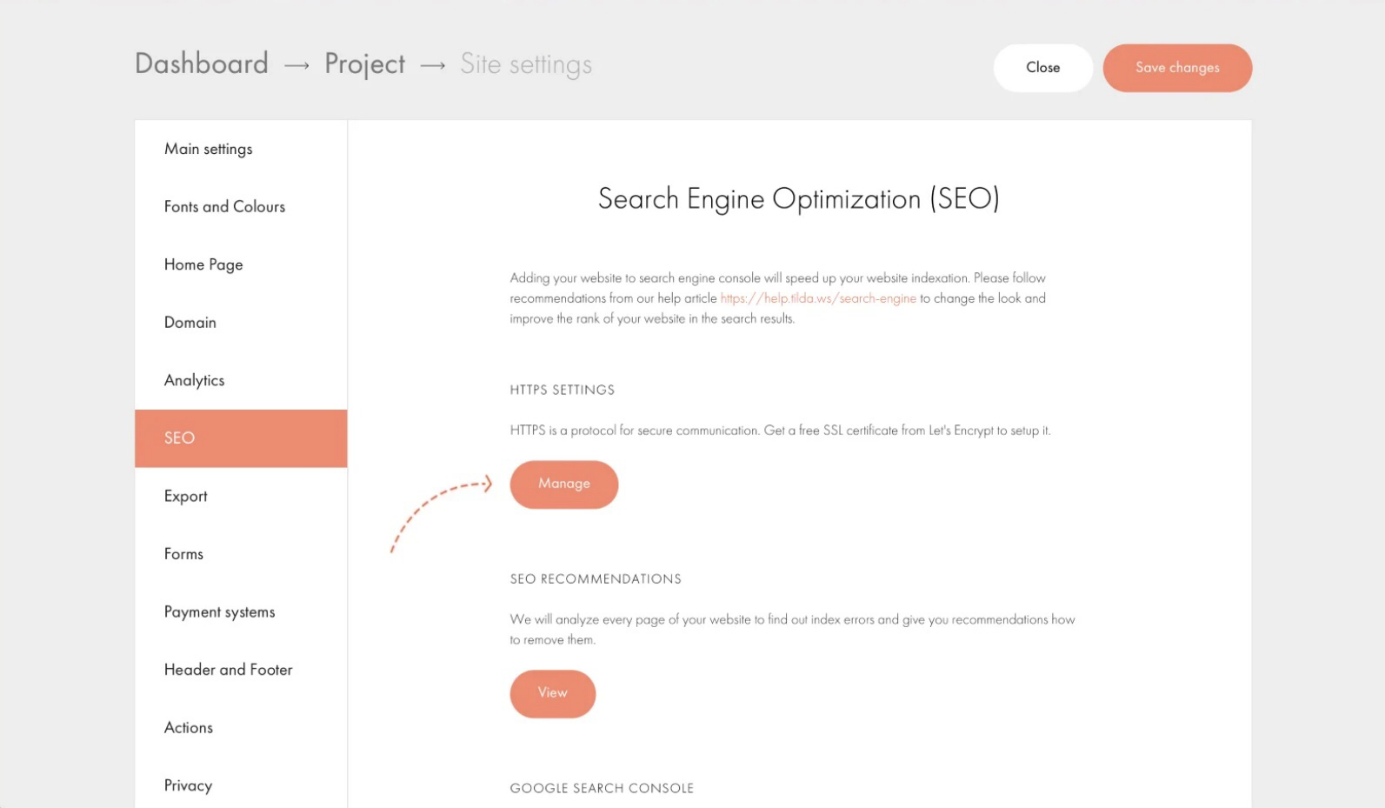


Image 6: Click Manage in HTTPS settings

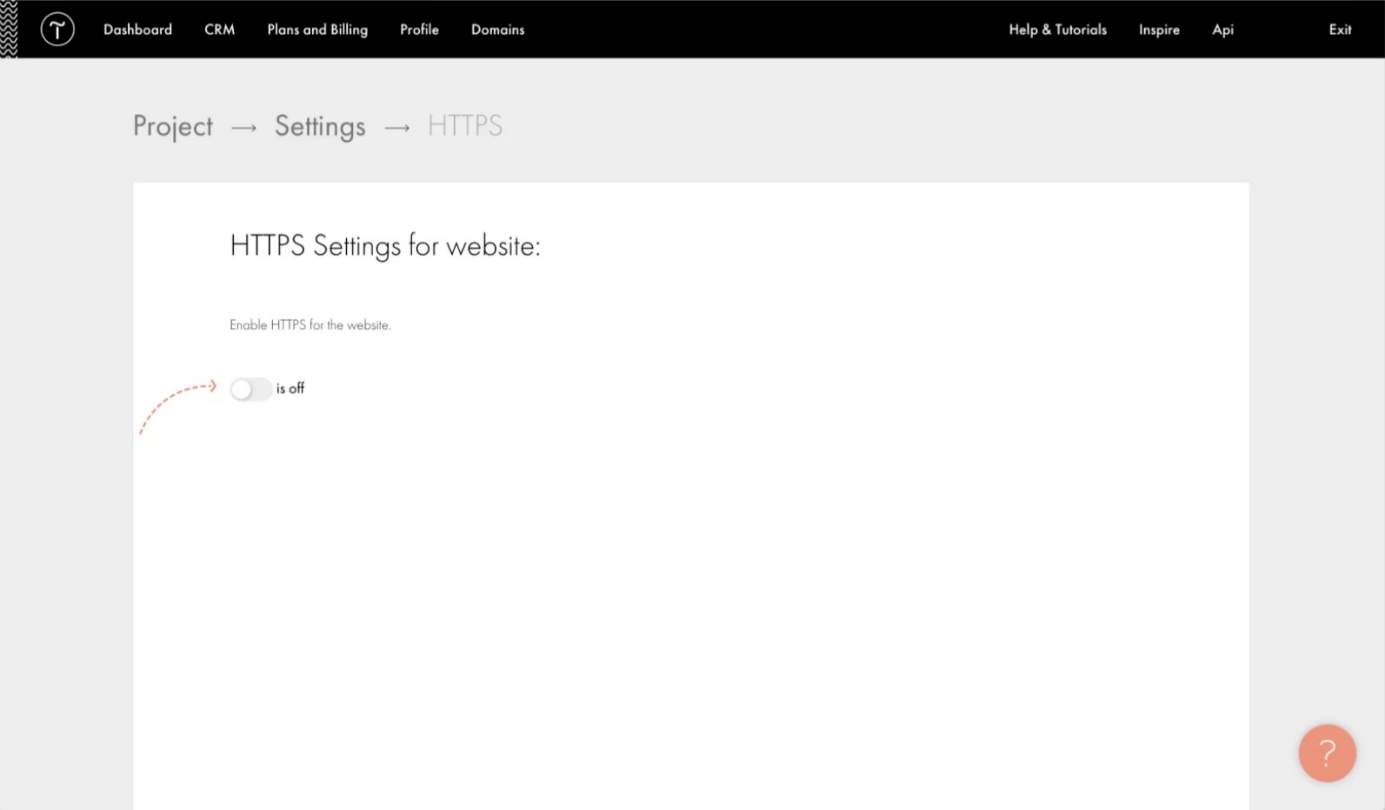


Image 7: Enable HTTPS for the website

1. Check the HTTPS version of the site.
2. Go to **Site Settings** → **SEO** → **Setup redirect from HTTP to HTTPS** to make the site opens only via HTTPS.

## Configuring HTTPS for a Nginx website

Before configuring HTTPS, you need to **buy and install an SSL certificate** on the website.

1. Сreate a **new configuration snippet** file to configure NGINX to use SSL:

sudo nano /etc/nginx/snippets/your\_new.conf

1.1. **Add** the following:

ssl\_certificate /etc/ssl/certs/your\_new.crt;

ssl\_certificate\_key /etc/ssl/private/your\_new.key;

1.2. **Save** and **close** the file.

1. Create **another configuration snippet** that points to our newly-generated SSL key and certificate:

sudo nano /etc/nginx/snippets/ssl-params.conf

2.1. **Add** the following:

ssl\_protocols TLSv1.2;

ssl\_prefer\_server\_ciphers on;

ssl\_dhparam /etc/ssl/certs/dhparam.pem;

ssl\_ciphers ECDHE-RSA-AES256-GCM-SHA512:DHE-RSA-AES256-GCM-SHA512:ECDHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384;

ssl\_ecdh\_curve secp384r1; # Requires nginx >= 1.1.0

ssl\_session\_timeout 10m;

ssl\_session\_cache shared:SSL:10m;

ssl\_session\_tickets off; # Requires nginx >= 1.5.9

ssl\_stapling on; # Requires nginx >= 1.3.7

ssl\_stapling\_verify on; # Requires nginx => 1.3.7

resolver 8.8.8.8 8.8.4.4 valid=300s;

resolver\_timeout 5s;

add\_header X-Frame-Options DENY;

add\_header X-Content-Type-Options nosniff;

add\_header X-XSS-Protection "1; mode=block";

If you are using a **self-signed** certificate, add the **# symbols** before lines 9 and 10 to disable SSL stapling:

# ssl\_stapling on; # Requires nginx >= 1.3.7

# ssl\_stapling\_verify on; # Requires nginx => 1.3.7

2.2. **Change the resolver line** to reflect your preferred DNS servers.

2.3. **Save** and **close** that file.

1. Generate the **dhparam.pem file**:

sudo openssl dhparam -out /etc/ssl/certs/dhparam.pem 2048

This command will take some time.

1. Configure NGINX so that it knows you are going to use SSL. If you have a **server block**, open it with the command:

sudo nano /etc/nginx/sites-available/example.com

4.1 In that file, **edit** it to reflect the following:

server {

listen 443 ssl;

listen [::]:443 ssl;

include snippets/your\_new.conf;

include snippets/your\_new.conf;

server\_name example.com www.example.com;

root /var/www/example.com/html;

index index.html index.htm index.nginx-debian.html;

}

4.2 Add a **new server block** **below** to perform an HTTPS redirect:

server {

listen 80;

listen [::]:80;

server\_name example.com www.example.com;

return 302 https://$server\_name$request\_uri;

}

1. Link from **sites-available to sites-enabled**:

ln -s /etc/nginx/sites-available/www.example.com /etc/nginx/sites-enabled/

1. **Restart** NGINX:

sudo systemctl restart nginx

# Summarized Information

The article covers such concepts as **HTTP** and **HTTPS certificates** and provides detailed instructions for **switching from HTTP to HTTPS** for websites created using a site builder called **Tilda**, and websites hosted on a server running **Nginx**.