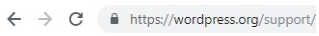
HTTPS

HTTPS is an encrypted communication protocol — essentially, a more secure way of browsing the web, since you get a private channel directly between your browser and the web server. That's why most major sites use it.

If a site's using HTTPS, you'll see a little padlock icon in the address field, just as in the screenshot below:



Here are the most common reasons you might want to use HTTPS on your own site:

Faster. One might think that HTTPS would make your site slower, since it takes some time to encrypt and decrypt all data. But a lot of efficiency improvements to HTTP are only available when you use HTTPS. As a result, HTTPS will actually make your site faster for almost all visitors.

Trust. Users find it easier to trust a secure site. While they don't necessarily know their traffic is encrypted, they do know the little padlock icon means a site cares about their privacy. Tech people will know that any servers between your computer and the web server won't be able to see the information flowing forth and back, and won't be able to change it.

Payment security. If you sell anything on your site, users want to know their payment information is secure. HTTPS, and the little padlock, assure that their information travels safely to the web server.

Search Engine Optimization. Many search engines will add a penalty to web sites that don't use HTTPS, thus making it harder to reach the best spots in search results.

Your good name. Have you noticed that some websites have the text "not secure" next to their address?

That happens when your web browser wants you to know a site is NOT using HTTPS. Browsers want you to think (rightly!) that site owners who can't be bothered using HTTPS (it's free in many cases) aren't worth your time and certainly not your money.

In turn, you don't want browsers suggesting you might be that kind of shady site owner yourself.

WordPress is fully compatible with HTTPS when an TLS / SSL certificate is installed and available for the web server to use. Support for HTTPS is strongly recommended to help maintain the security of both WordPress logins and site visitors.

Administration Over HTTPS

To easily enable (and enforce) WordPress administration over SSL, there are two constants that you can define in your site's wp-config.php file. It is not sufficient to define these constants in a plugin file; they must be defined in your wp-config.php file. You must also already have SSL configured on the server and a (virtual) host configured for the secure server before your site will work properly with these constants set to true.

Note: `FORCE_SSL_LOGIN` was deprecated in Version 4.0. Please use `FORCE_SSL_ADMIN`.

To Force HTTPS Logins and HTTPS Admin Access

The constant `FORCE_SSL_ADMIN` can be set to true in the `wp-config.php` file to force all logins and all admin sessions to happen over SSL.

Example

```
define( 'FORCE_SSL_ADMIN', true );
```

Using a Reverse Proxy

If WordPress is hosted behind a reverse proxy that provides SSL, but is hosted itself without SSL, these options will initially send any requests into an infinite redirect loop. To avoid this, you may configure WordPress to recognize the `HTTP_X_FORWARDED_PROTO` header (assuming you have properly configured the reverse proxy to set that header).

Example

```
define( 'FORCE_SSL_ADMIN', true );
// in some setups HTTP_X_FORWARDED_PROTO might contain
// a comma-separated list e.g. http,https
// so check for https existence
if( strpos( $_SERVER['HTTP_X_FORWARDED_PROTO'], 'https') !== false )
    $_SERVER['HTTPS'] = 'on';
```

Further Information

Further Information

The rest of this article serves as information in case you're using an older version of WordPress (which ideally you shouldn't!) or your SSL setup is somewhat different (ie. your SSL certificate is for a different domain).

Sometimes, you want your whole wp-admin to run over a secure connection using the https protocol. Conceptually, the procedure works like this:

1. Set up two virtual hosts with the same url (the blog url), one secure, the other not.
2. On the secure virtual host, set up a rewrite rule that shuttles all non-wp-admin traffic to the insecure site.
3. On the insecure virtual host, set up a rewrite rule that shuttles all traffic to wp-admin to the secure host.
4. Put in a filter (via a plugin) that filters the links in wp-admin so that once activated, administrative links are rewritten to use https and that edits cookies to work only over encrypted connections.

The following guide is for WordPress 1.5 and Apache running `mod_rewrite`, using rewrite rules in `httpd.conf` (as opposed to `.htaccess` files) but could easily be modified to fit other hosting scenarios.

Virtual Hosts

You need a (virtual) host configured for the secure server in addition to the non-secure site. In this example, the secure virtual host uses the same `DocumentRoot` as the insecure host. Hypothetically, you could use a host with a different name, such as wpadmin.mysite.com and link the document root to the wpadmin directory.

Please ask your ISP to set up a secure virtual host for you, or if you have administrative access set up your own. Note that you cannot use name based virtual hosting to identify different SSL servers.

Rewrite Rules For The Insecure Host

In the `.htaccess` or virtual host stanza in `httpd.conf` for your insecure host, add this rewrite rule to automatically go to the secure host when you browse to https://example.com/wp-admin/ or https://example.com/wp-login.php

This should go above the main wordpress rewrite block.

```
RewriteCond %{THE_REQUEST} ^[A-Z]{3,9}\ /(.*)\ HTTP/ [NC]
RewriteCond %{HTTPS} !=on [NC]
RewriteRule ^/?(wp-admin/|wp-login\.php) https://example.com%{REQUEST_URI}%{QUERY_STRING} [R=3
01,QSA,L]
```

If you are using permalink rewrite rules, this line must come before `RewriteRule ^.*$ - [S=40]`.

An important idea in this block is using `THE_REQUEST`, which ensures only actual http requests are rewritten and not local direct file requests, like an include or fopen.

Rewrite Rules For Secure Host (Optional)

These rewrite rules are optional. They disable access to the public site over a secure connection. If you wish to remain logged in to the public portion of your site using the plugin below, you must not add these rules, as the plugin disables the cookie over unencrypted connections.

The secure virtual host should have two rewrite rules in an .htaccess file or in the virtual host declaration (see Using Permalinks for more on rewriting):

```
RewriteRule !^/wp-admin/(.*) - [C]
RewriteRule ^/(.*) https://www.example.com/$1 [QSA,L]
```

The first rule excludes the wp-admin directory from the next rule, which shuffles traffic to the secure site over to the insecure site, to keep things nice and seamless for your audience.

Setting WordPress URI

For some plugins to work, and for other reasons, you may wish to set your WordPress URI in options to reflect the https protocol by making this setting https://example.com. Your blog address should not change.

Example Config Stanzas

NOTE: The below config is not 100% compatible with WordPress 2.8+, WordPress 2.8 uses some files from the wp-includes folder. The redirection that the first set of Rewrite rules introduces may cause security warnings for some users. See #10079 for more information.

```
<VirtualHost nnn.nnn.nnn.nnn:443>
    ServerName www.example.com

    SSLEngine On
    SSLCertificateFile /etc/apache2/ssl/thissite.crt
    SSLCertificateKeyFile /etc/apache2/ssl/thissite.pem
    SetEnvIf User-Agent ".*MSIE.*" nokeepalive ssl-unclean-shutdown

    DocumentRoot /var/www/mysite

    <IfModule mod_rewrite.c>
        RewriteEngine On
        RewriteRule !^/wp-(admin|includes)/(.*) - [C]
        RewriteRule ^/(.*) https://www.example.com/$1 [QSA,L]
    </IfModule>

</VirtualHost>
```

Insecure site

```
<VirtualHost *>
    ServerName www.mysite.com

    DocumentRoot /var/www/ii/mysite

    <Directory /var/www/ii/mysite >
        <IfModule mod_rewrite.c>
            RewriteEngine On
            RewriteBase /
            RewriteCond %{REQUEST_FILENAME} -f [OR]
            RewriteCond %{REQUEST_FILENAME} -d
            RewriteRule ^wp-admin/(.*) https://www.example.com/wp-admin/$1 [C]
            RewriteRule ^.*$ - [S=40]
            RewriteRule ^feed/(feed|rdf|rss|rss2|atom)/?$ /index.php?&feed=$1 [QSA,L]

        </IfModule>
    </Directory>

</VirtualHost>
```

Rewrite for Login and Registration

It is probably a good idea to utilize SSL for user logins and registrations. Consider the following substitute RewriteRules.

Insecure

```
RewriteRule ^/wp-(admin|login|register)(.*) https://www.example.com/wp-$1$2 [C]
```

Secure

```
RewriteRule !^/wp-(admin|login|register)(.*) - [C]
```

Rewrite for sites running on port 443 or port 80

```
# BEGIN WordPress
<IfModule mod_rewrite.c>
RewriteEngine On
RewriteBase /

# For a site running on port 443 or else (http over ssl)
RewriteCond %{SERVER_PORT} !^80$
RewriteRule !^wp-(admin|login|register)(.*) - [C]
RewriteRule ^(.*)$ https://%{SERVER_NAME}/$1 [L]

# For a site running on port 80 (http)
RewriteCond %{SERVER_PORT}  ^80$
RewriteCond %{REQUEST_FILENAME} -f [OR]
RewriteCond %{REQUEST_FILENAME} -d
RewriteRule ^wp-(admin|login|register)(.*) https://%{SERVER_NAME}:10001/wp-$1$2 [L]

RewriteCond %{SERVER_PORT}  ^80$
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule . /index.php [L]

</IfModule>
```

Summary

This method does not fix some inherent security risks in WordPress, nor does it protect you against man-in-the-middle attacks

This method does not fix some inherent security risks in WordPress, nor does it protect you against man-in-the-middle attacks or other risks that can cripple secure connections.

However, this should make it much harder for a malicious person to steal your cookies and/or authentication headers and use them to impersonate you and gain access to wp-admin. It also obfuscates the ability to sniff your content, which could be important for legal blogs which may have drafts of documents that need strict protection.

Verification

On the author's server, logs indicate that both GET and POST requests are over SSL and that all traffic to wp-admin on the insecure host is being shuttled over to the secure host.

Sample POST log line:

```
[Thu Apr 28 09:34:33 2005]

 Subsequent (No.5) HTTPS request received for child 6 (server foo.com:443)
xx.xxx.xxx.xxx - - [28/Apr/2005:09:34:33 -0500] "POST /wp-admin/post.php HTTP/1.1" 302 - "http
s://foo.com/wp-admin/post.php?action=edit&post=71" "Mozilla/5.0 (Windows; U; Windows NT 5.1; e
n-US; rv:1.7.7) Gecko/20050414 Firefox/1.0.3"
```

More testing, preferably with a packet sniffer and some hardcore network analysis tools, would help to confirm.

Limitations

The author assumes (but hasn't checked) that if the user has stored cookies/told their browser to remember passwords (not based on form fields but if using certain external auth mechanism) and hits https://www.example.com/wp-admin/, those packets are sent in the clear and the cookie/auth headers could be intercepted. Therefore, to ensure maximum security, the user should explicitly use the https host or always log in at the beginning of new sessions.

Changelog

- 2022-10-25: Original content from Why should I use HTTPS, and Administration Over SSL.

First published

March 28, 2023

Last updated

January 16, 2024

Previous
Backing Up Your WordPress Files

Next
Brute Force Attacks