Trusted Host settings

drupal.org/docs/8/install/trusted-host-settings

Protecting against HTTP HOST Header attacks (prevent your site from thinking it is someone else)

Drupal 7 added a new feature into core that is not user facing directly, but is sometimes called poor man's cron. The feature triggers the periodic tasks of a Drupal site like emptying log files, sending e-mails, and clearing out caches. This feature, when combined with dynamic detection of the "base url" (added in Drupal 4.7), can lead to some screwy situations. This article is a description of some of those screwy situations that occur with either module or both of them and what you can do to prevent them. The comments below assume some default configurations - I'll discuss at the end how to break away from those defaults to prevent these problems.

Scenario 1: Getting/sending user emails that appear to be for another domain

This behavior is pretty easy to replicate:

The result is that in step 2 the $base_url detection thinks that your site is http://other-site.example.org and all the tokens for the e-mail like [user:one-time-login-url]that contain links to your site will be changed to use http://other-site.example.org as the base url. The user who receives this email will see their username and e-mail for example.com are now somehow in use on http://other-site.example.org, which is usually just confusing. Two bad scenarios could come from this, though:

- In a worst-case scenario could lead to them click on the password reset link which the evil site could then use to login to the site as that user
- They might enter their username/password into http://other-site.example.org - a so-called social engineering attack - which could then be used on the main site

Scenario 2: Cache entries containing the wrong domain

A similar problem can occur when a user uses the wrong domain to make a request and that happens to be the request that primes a cache entry with dynamic, fully qualified domains in it. Subsequent visits that retrieve information from that cache will get the wrong domain name. Drupal core's page cache uses the domain as part of the cache ID, preventing this problem, but other caching mechanisms may not be as robust against this problem.

Scenario 3: Notification mails containing the wrong domain

Yet another problem can occur on sites that use modules that send e-mails during cron runs. This scenario requires the poor-man's-cron with the dynamic base_url detection. If a user happens to trigger the poor man's cron when there are notifications in the queue by

visiting the wrong domain name then notifications will be sent with that incorrect domain. Users will be very confused about why the mail they expect to receive coming from an e-mail address at example.com includes links to the http://other-site.example.org domain.

Solutions to confusing experiences with Drupal's dynamic base_url detection

There are at least four potential solutions to this problem, though not all are necessary to stop the problem from happening. You should pick and choose as appropriate for your environment.

1. You can set a specific domain as your $base_url in sites/default/settings.php. While the dynamic detection can be a handy feature it can also cause problems. One way to stop that is just to set a permanent value.
2. Use a specific sites/example.com/settings.php and let $base_url be detected dynamically - this has the implication of letting Drupal respond to all subdomains of example.com which may or may not be a benefit.
3. Configure your webserver so that the default page served when an incoming request is something other than your default Drupal installation, such as an error page.
4. Configure your webserver to redirect all requests that reach your server *that are not for the appropriate domain* to forward to the right domain name.

Trusted host security setting in Drupal 8

As of January 2015, Drupal 8 supports "trusted host patterns", where you can (and should) specify a set of regular expressions that the domains on incoming requests *must* match. Example configuration in `settings.php` would read:

```
$settings['trusted_host_patterns'] = [
  '^www\.example\.com$',
];
```

See the above change record for more details. Note that, if you're doing local development, you might get (temporarily) locked out of your site by the above configuration on its own. You should add a trusted host pattern for `'^localhost$'` in this case.

MAMP possible exception

About local development, MAMP (3.5.2) '^localhost$' setting give the error message "The provided host name is not valid for this server" and doesn't load the site. Found a solution changing it with site name without port number. In my test site "drupal8":

```
$settings['trusted_host_patterns'] = [
  '^drupal8$',
];
```

made Trusted Host active.

NOTE: On MAMP 4.2 '^localhost$' works just fine.

Trusted host security setting in Drupal 8.4.0 and PHP 7.1.8 for XAMP

To enable the trusted host mechanism, we need to enable our allowable hosts in $settings['trusted_host_patterns'].

Open the "settings.php" file and update the below code to Enable the Trusted host setting:

```
$settings['trusted_host_patterns'] = array(
'^localhost$',
'^192\.168\.00\.52$',
'^127\.0\.0\.1$',
);
```

Here,

- '^localhost$', : This will allow the site to only run from localhost.
- '^192\.168\.00\.52$', : This will allow the site to only run from system IP (different system has different IP).
- '^127\.0\.0\.1$', : This will allow the site to only run from 127.0.0.1 instead of localhost.

Note : If someone running multisite, then in that case just specify all of the host patterns that are allowed by the site.