**HTTP HOST HEADER INJECTION**

Host header injections been around for a while now, and sometimes developer just don't know how to get rid of them! Configuring virtual hosts, adding host verification codes to redirection pages etc. All of this can be time consuming or stressful for some!

So, I will be explaining about a simple technique that developers can use to mitigate all the issues related to host header manipulation.

Let's start with basic understanding of Host header injection.

So in simple words, Host header injection is to change the value of Host header in the request to any other domain. Then server uses the modified Host value in common tasks like redirection links, sending emails, password reset links etc. which can lead to variety of attacks.

Another possible injection technique for Host headers can be through X-Forwarded-Host header. In some configurations this header might rewrite the value of Host header.

This is enough overview about the attack possibilities! If you do like to read more then I've put some good links in the references below.

Now let's see what we can do to protect against these attacks.

**Apache**

For Apache and it’s variants the best way to mitigate Host Header Injection is by creating Virtual Hosts entries in the configuration file (httpd.conf). If multiple domains/sub-domains are hosted on the same server, then multiple vhost entries can be created as shown in the below snippet.

<VirtualHost \*:80> #Configures the first and default virtual host listening for ALL IPs on PORT 80. It is the default because it is the first.

       ServerName yourdomain.com

       ServerAlias www.yourdomain.com

       UseCanonicalName On # Forces apache to use ServerName mentioned above instead of HostName from client request.

       DocumentRoot /path/to/your/webroot/directory

</VirtualHost>

<VirtualHost \*:80>   #Configures the second virtual host. [For second domain/ subdomain hosted on the same server (if any)].

       ServerName test1.yourdomain.com

       UseCanonicalName On

       DocumentRoot /path/to/your/webroot/directory/of/second/domain

</VirtualHost>

As an alternative for Apache and it’s variants I recently created a redirect rule that can be put in .htaccess or in httpd.conf files. These rules work based on a regular expression that will verify the value of Host header and if the value does not match the regex pattern then a redirection is performed to a predefined URL.

Note: ^([a-zA-Z0-9-\_]{1,20}.){0,3}domain.com$ will allow subdomains till 3rd level and each subdomain name can be between 1 to 20 character length. If you have N levels of subdomains, then replace {0,3} with {0,N}. If subdomain name length is X character long (more than 20), then replace {1,20} with {1,X}.

RewriteEngine On

RewriteCond %{HTTP\_HOST} !^([a-zA-Z0-9-\_]{1,20}.){0,3}domain.com$

RewriteRule ^(.\*)$ https://domain.com/ [R=301,L]

The above code snippet checks if the Host header is equal to domain.com or any subdomain of domain.com. If the Host header mismatches, then a 301 redirection to https://domain.com/ is performed.

Above is the case when the server is having only one main domain hosted. If there are multiple domains, then the following snippet can be used.

RewriteEngine On

RewriteCond %{HTTP\_HOST} !^([a-zA-Z0-9-]{1,20}.){0,3}(domain1.com|domain2.com|domain3.com)$

RewriteRule ^(.\*)$ https://domain1.com/ [R=301,L]

The above code snippet will match the Host value with domain1.com, domain2.com, domain3.com and all their subdomains. If mismatched, a 301 redirection response for domain1.com will be sent to the user. More domains can be added to this rule in a similar fashion.

Now, suppose X-Forwarded-Host header is used on the server. Then since these forwards are done by the server-side implemented technologies, the developers already know their addresses. So just like the above defined regex-based rules, a new rule can be set to check if this header is present with a valid address.

RewriteEngine On

RewriteCond %{HTTP:X-Forwarded-Host} !^$

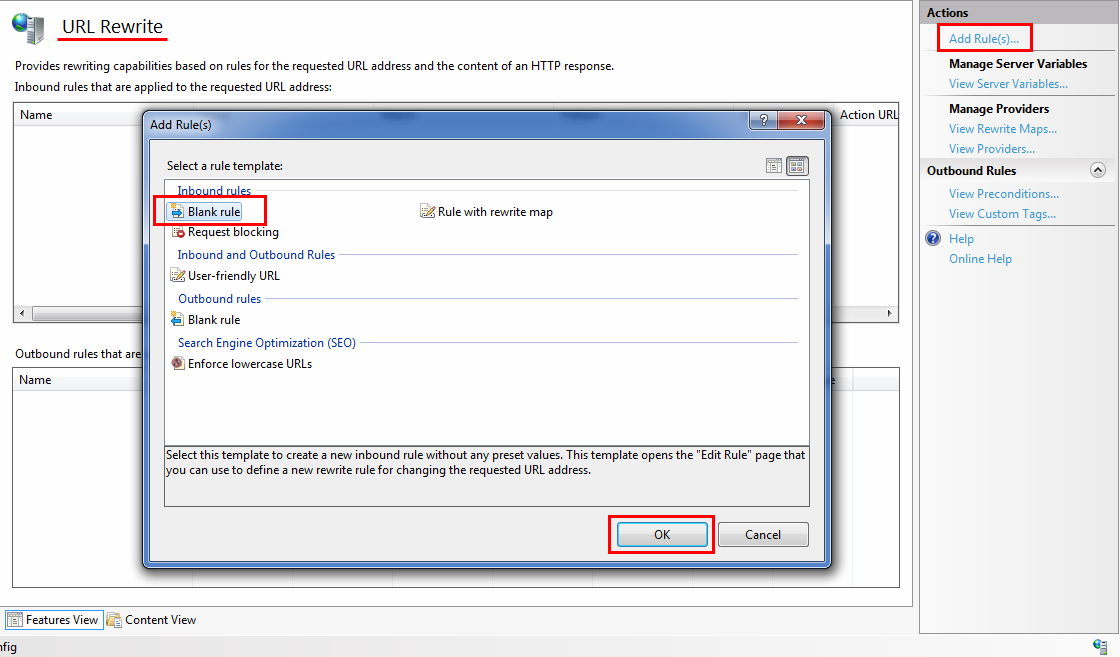
RewriteCond %{HTTP:X-Forwarded-Host} !^([a-zA-Z0-9-\_]{1,20}.){0,3}(domain1.com|domain2.com|domain3.com)$

RewriteRule ^(.\*)$ https://domain1.com/ [R=301,L]

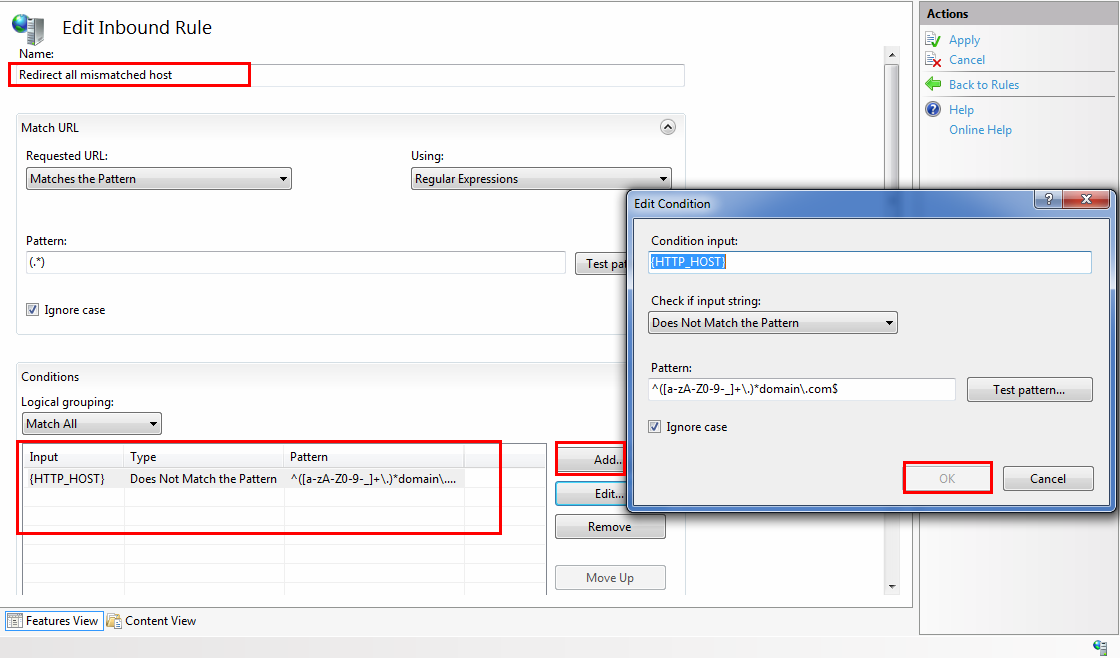
The above snippet checks if the X-Forwarded-Host header is present. Then checks if it matches the allowed addresses. In case of a mismatch, a 301 redirection is performed to domain1.com.

**IIS**

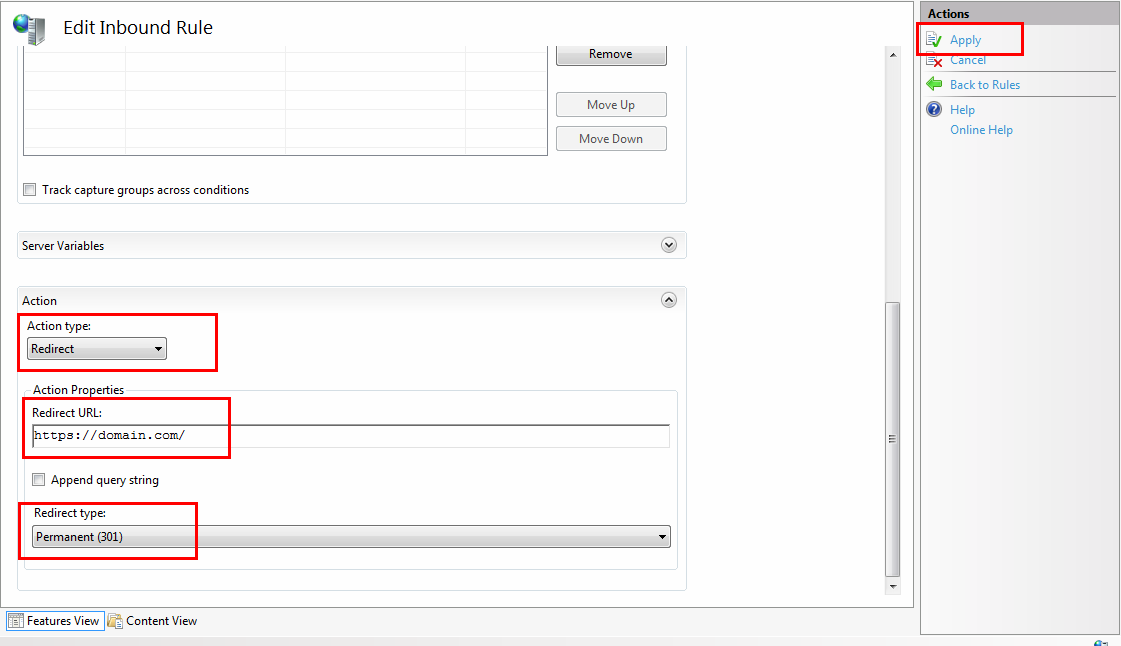
Above regex patterns can also be implemented in IIS, via URLRedirect module. Follow the below steps:



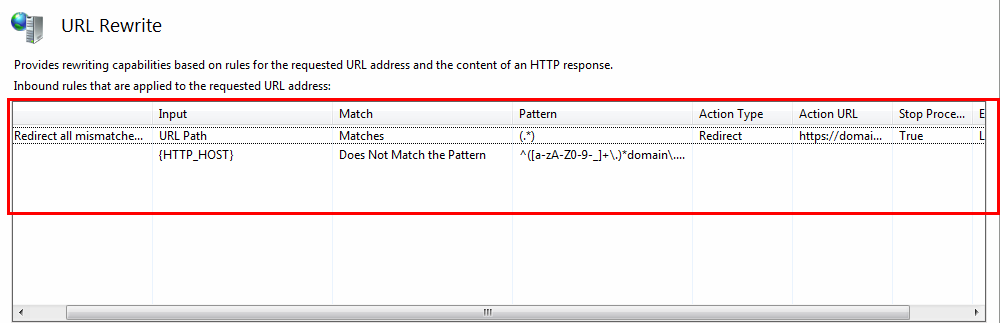
1. Open URL Rewrite module. Then click on Add Rules and create a blank rule.



2. Give a name to this rule, so that it can be identified later. In Match URL section set the pattern to (.\*) which means everything. So this rule will be implemented on all URLs. Then in conditions section click on Add button. Then Condition input is {HTTP\_HOST}, check is "does not match the pattern" and the pattern is ^([a-zA-Z0-9-\_]+\.)\*domain\.com$

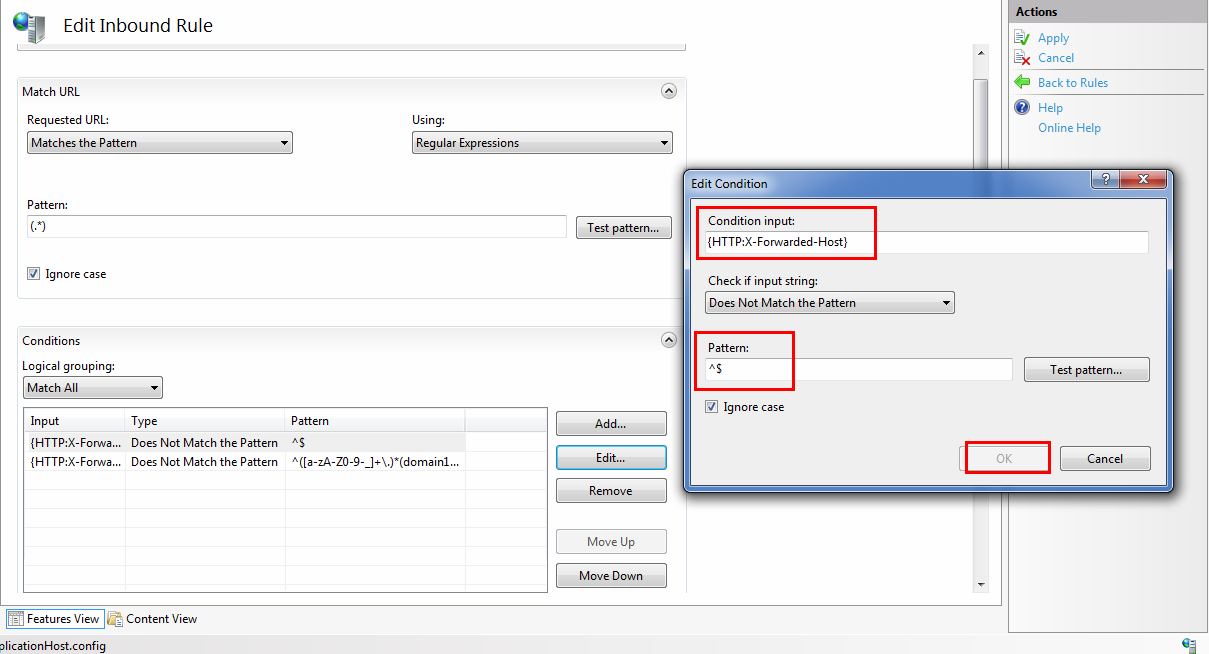


3. Now in Action section, type is redirect, URL is https://domain.com/ and redirection type is 301. Then click on

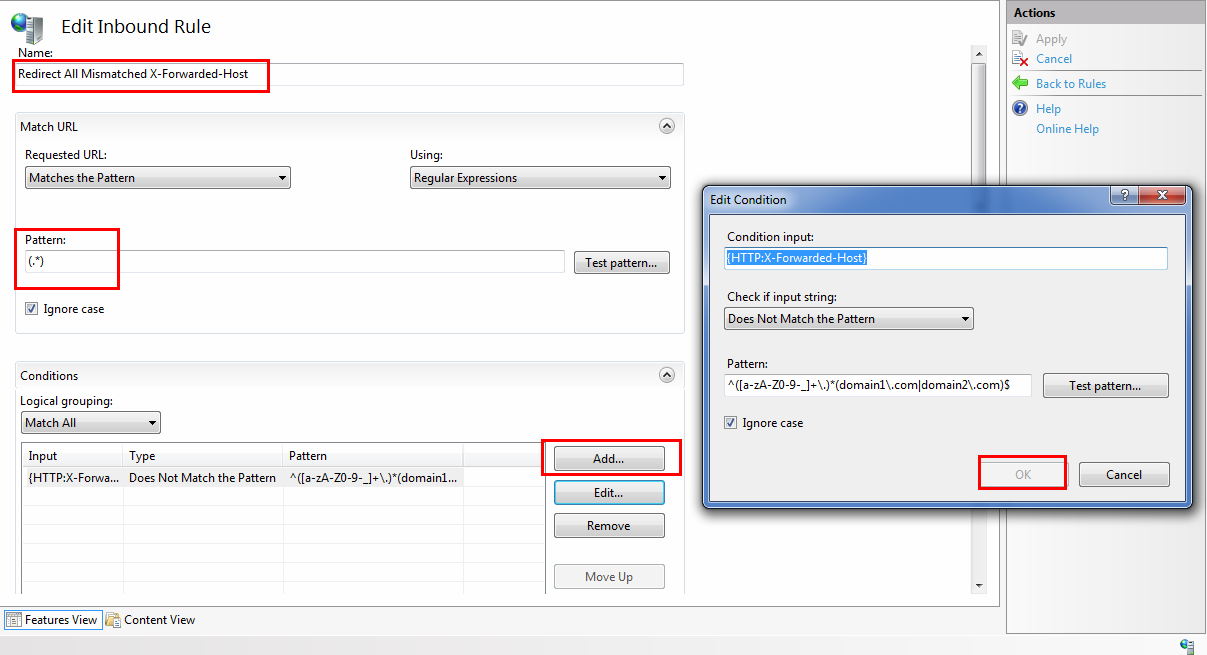


4. Final rule looks like this. In case, this rules needs to be implemented for multiple domains, then regex pattern will be ^([a-zA-Z0-9-\_]+\.)\*(domain1\.com|domain2\.com|domain3\.com)$

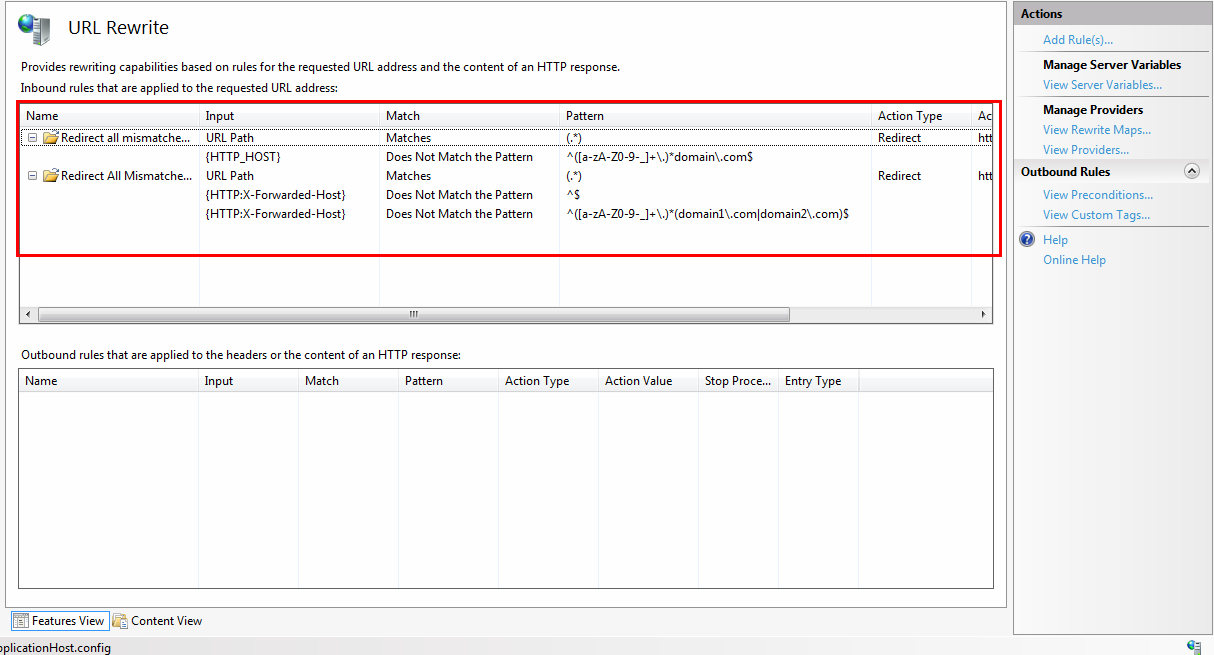
For X-Forwarded-Host the rule will be set like this:



1. Condition input will be {HTTP:X-Forwarded-Host}, check will be "Does not match the pattern" and Pattern will be ^$



2. For second condition, condition input will be {HTTP:X-Forwarded-Host}, check will be "Does not match the pattern" and Pattern will be ^([a-zA-Z0-9-\_]+\.)\*(domain1\.com|domain2\.com|domain3\.com)$



3. Finally rules will look like this.

That's all what's needed to protect your application from Host manipulations.

There is much more potential in redirect rules and these were just a few demonstrations. You guys can surely use them as it is or modify them according to the need. If there is any query feel free to comment or message me. Always open for suggestions :)

**References**

1) <http://skeletonscribe.net/2013/05/practical-http-host-header-attacks.html/>

2) <http://carlos.bueno.org/2008/06/host-header-injection.html>

3) [https://www.silverstripe.org/download/security-releases/ss-2015-013/](http://hackerone.com/redirect?signature=673774231857c512e368b6e4de010141a4faca86&url=https%3A%2F%2Fwww.silverstripe.org%2Fdownload%2Fsecurity-releases%2Fss-2015-013%2F)

4) [https://www.keycdn.com/support/x-forwarded-host/](http://www.keycdn.com/support/x-forwarded-host/)

5) <https://technet.microsoft.com/en-us/library/cc753195(v=ws.10).aspx>