

Forum:[Hack Forums](#)

Thread: [\[Infamous/Antagonism\] Bypassing .htaccess/.htpasswd Based Authentication Systems](#)

Keeper - April 8, 2013, 8:21 am

Little off-topic:

It's been some time since I posted some decent stuff related to hacking and as a whole in this section. Most of my recent activity was aimed at the PHP development section and mainly coding.

Introduction:

Though, the title is not entirely correct, the actual authentication method is called BA (abbreviated for Basic Access). The form itself requests an HTTP user-agent to supply the credentials. Also there are no active sessions established during your activity within the system. If you still don't know what I'm talking about take a look at the image I included at the beginning (deliberately).

Exploitation:

Basically, I could just skip all the redundant information and briefly describe how to exploit it but the idea is different. I want the members who read this to gain the actual logic of how the things are being processed and how the whole scheme is being organised. First off, imagine we have the following .htaccess rules:

```
Code AuthUserFile /mnt/web/guide/somewhere/somepath/.htpasswd
AuthGroupFile /dev/null
AuthName [Directory we want to restrict access to]
AuthType Basic
```

The above rules enable the most basic, yet successful logging. The URL that is being parsed to the server is being handled by mod_php. What PHP does is render the request method, either as HEAD or POST. In the case of sending the method as "GETS", that leads to the invalid method used to send the payload not being checked further (since PHP does not validate that) and resulting in the standard output of the shell. Therefore, the status code that we receive is equal to 200 which pretty much means the request has been successful.

Now in order to escape such cases, we need to define the limit of methods that are permitted to be used for the request, denying everything except the two methods - POST & GET.

```
Code <LimitExcept GET POST>
Order Allow,Deny
Deny from all
</LimitExcept>
```

The part of the actual exploitation pretty much consists of replaying the request to the directory that is using the authentication system. You could do that in numerous ways, but for the sake of this tutorial, I'm going to go through two of them. The easier one is using the add-on Live HTTP Headers , which allows you to view or/and modify the headers in the request and the other one is using a PHP-based script for opening/establishing a raw socket connection to the specified server.

Using HTTP Live Headers, just specify the method (which is ought to be different than POST or GET) and replay the request beforehand so that you get enabled to modify that.

Once this is achieved, you will automatically get authorized without any supplied credentials to the form. The other way round this, that I'm going to explain is the already mentioned socket connection.

PHP Code

```
<?php
error_reporting(0);

$request = "Keeper / HTTP/1.1\r\n
User-Agent: Mozilla/4.0 (Compatible; whatever)\r\n
Host: google.com\r\n
Accept: */*\r\n";

$urltarget = $_POST['url'];

$socket = fsockopen($urltarget);
fwrite($socket,$request);
$received = fread($socket,1024*1024);
echo $received;

?>
```

The socket establishes a connection to the target and returns the status of the request for confirmatory reason. Firstly, we define the headers under the variable \$request. Then we specify that the target URL should be a POST value of a form supplied input (you can put the three lines of HTML in order to visualize that or neglect the usage of \$_POST and just use the URL directly). Then we establish the actual connection with the function fsockopen() and fwrite(). We store the result under the variable \$received which is receiving the request with 1MB of size (just to make sure we receive the whole of it).

And let's actually see through telnet how our method is processed.

Using "GETS" as method we receive the 501 status code which means that the server either does not recognize the request method, or it lacks the ability to fulfill it thus getting us authorized.

Conclusion:

That's what I wanted to share. Of course, we could've gone much deeper but the point is to understand the basic scheme of the whole process, not just to implement random queries, vectors, payloads or what not (as is the case of SQLi and XSS..).

'Snorlax - April 8, 2013, 8:30 am

Wow I never looked at it this way.

I was just using HTTP Live Headers today actually, so thanks for sharing this.

Interesting read.

LevelUp - April 8, 2013, 8:48 am

Sockets was a really nice addition to it, nice tutorial bro.

Even nicer site you've tried it on :3

Keeper - April 8, 2013, 8:55 am

(Apr 8, 2013 08:52 AM)ankr- Wrote: [\[CITE\]](#) Nice tut, though you suc kat censoring IP's
Oops.. Edited that. Thanks for the note.

pashabiceps - April 8, 2013, 8:56 am

^ Haha

Amazing tutorial. Keep them coming!

-?Divine- - April 8, 2013, 9:19 am

Interesting read. Good job Keeper.

(Apr 8, 2013 08:52 AM)ankr- Wrote: [\[CITE\]](#) Nice tut, though you suc kat censoring IPs
Spoiler [\(Click to View\)](#)-removeeedd-

selling the IP, PM me, 500\$, you go first no refunds, no warranty

I GOT THE IP AND I'M SELLING IT FOR ONLY \$499.99!

amus3d - April 8, 2013, 9:35 am

You put a lot of time and effort into this. :3

Gewd job, looks very interesting.

-Elusive- - April 8, 2013, 10:43 am

Great tutorial! Thanks for sharing :)

h?cker22 - April 8, 2013, 10:45 am

Good job. You're a great asset to the community, Keeper. ?

xx 0x0 xx - April 8, 2013, 9:33 pm

one of the best HQ tutorials at HF nice work dude i really like this tut gonna help me alot never heard of this before

Keeper - April 9, 2013, 9:56 am

(Apr 8, 2013 09:33 PM)mats0o0o0 Wrote: [\[CITE\]](#) one of the best HQ tutorials at HF
Uhm, no? Anyway, thanks for the comments

whitehatreborn - April 10, 2013, 2:57 am

Does anyone have a dork 4 this? You could atleast provide a test site.

Keeper - April 10, 2013, 3:53 am

(Apr 9, 2013 02:12 PM)ankr- Wrote: [\[CITE\]](#) OK if you are so confident about this, then show me ONE better tutorial than this one here on HF.
ha 0wned skid.

You can check some of my other tutorials. I don't consider this one better than any of my others.

(Apr 10, 2013 02:57 AM)whitehatreborn Wrote: [\[CITE\]](#) Does anyone have a dork 4 this? You could atleast provide a test site.

Oh lookie what we have here. You want dorks so that you may implement it on random websites? Just setup your own environment, it takes no more than 5 minutes.

thesnipair.24 - April 10, 2013, 6:17 am

I've searched for this before like 4 years and couldn't do it, even wasn't thinking about this method thanks a lot great tutorial.

TheNeutron - April 10, 2013, 12:53 pm

Interesting tutorial mate... Thank you :)

Keep sharing please :D

Keeper - April 20, 2013, 7:21 am

I'll post another one on this topic soon.

Mormaii - April 20, 2013, 2:21 pm

This tutorial is old as fuck. This method has been known for ages.

Keeper - April 20, 2013, 2:28 pm

(Apr 20, 2013 02:21 PM)Mormaii Wrote: [\[CITE\]](#) This tutorial is old as fuck. This method has been known for ages.

Yes, but this is HF. Most people don't know about it and is useful for lame written .htaccess rules.

thesnipair.24 - June 14, 2013, 8:15 pm

(Apr 20, 2013 02:28 PM)Keeper Wrote: [\[CITE\]](#) Yes, but this is HF. Most people don't know about it and is useful for lame written .htaccess rules.

Is there a new way of bypassing it ?

DarkElement - June 15, 2013, 2:55 am

Nice looks good thought i use a simpler method (BT5R3) this looks good!

alvinthugz - June 18, 2013, 12:35 am

great job bro. keep it up :)

btw thanks for this :)

CatNips - July 1, 2013, 10:43 am

You put a lot of time and effort into this. :3

Gewd job, looks very interesting.