

Forum:[Hack Forums](#)

Thread: [\[Tutorial\] Morphing XSS attacks via \\$_REQUEST](#)

Th3PonyWizard - August 15, 2014, 10:00 am

Hello. In this (very) brief thread I'm going to explain the following term: XSS Morphing, and one technique of performing it.

Our target:

URL: <http://hsld-clan.com/XSSMorph/index.php>

XSS Morphing:

Is the term used to change one type of XSS into another by several common, less common, or completely private tricks. From Self-XSS to DoM XSS, to Reflected XSS to maybe even Stored XSS, it's all possible one way or another. If I'll care enough I'll reveal one of my actual half interesting ones. This is kindergarten level ;^)

We're going to attempt a basic XSS attack pattern. First, we'll view the source code:

Code <html>

<head>

<title>xss</title>

</head>

<body>

<form action="" method="post">

Write something:<input type="text" name="input">

<input type="submit" name="go">

</form>

</body>

</html>Alright, so, it appears our target input works through POST. It is a shame as we can't link a POST based request to our target in hopes he'll click on it and give us his delicious, moist cookies. However, we have come this far and we're not ready to give up yet.

Lets attempt an attack Proof Of Concept:

It works. What now?

It's simple, we take the parameter and shove it as a GET one.

Like this: Code <http://hsld-clan.com/XSSMorph/index.php?input=a&go=Submit+Query>

Wait, Sonder, How the fuck did it happen? Our form is clearly POST oriented, meaning it only accepts POST method-based input and displays it rendering XSS as useless!

No, my sweet summer child. You see, our silly pony programmer used the \$_REQUEST methodology instead of \$_POST.

Details: <http://php.net/manual/en/reserved.variables.request.php>

Basically, \$_REQUEST is a superglobal PHP array (Just like \$_GET and \$_POST), but it includes both of them and \$_COOKIE as well, making whatever you're doing in HTML pointless when your

PHP serverside script accepts all input no matter it's method. I would honestly suggest not to use `$_REQUEST` at all price because of that one rare case when you forget to filter the output of the registered username, but it's not that dangerous because of the POST request done, right? Yeah, if you're using `$_REQUEST` you're screwed.

The ways to handle against this are simple:

- 1) Learn to filter variables. This is shown at my White-Hat tutorial, look it up.
- 2) Remember our "go" named submit button? It's also working through `$_REQUEST`. What if we forced it to use `$_POST`? Then the XSS will go to shit and won't work as you can only verify the form being sent through POST.

The source code of my page, for future reference:

PHP Code

```
<?php
<html>
<head>
<title>xss</title>
</head>
<body>
<form action="" method="post">
Write something:<input type="text" name="input">
<input type="submit" name="go">
</form>
</body>
</html>
<?php
$input = $_REQUEST['input'];
if(isset($_REQUEST['go'])) {
echo $input;
}
?>
```

Note: This tutorial's writer is participating in The Sonder Challenge, forcing him to write a silly tutorial per day until he runs out of ideas. This is my first tutorial in the challenge. New one tomorrow. Challenge link: <https://hackforums.net/showthread.php?tid=4395018>
I'm out.

Aura - August 15, 2014, 10:44 am

Tell me why you aren't an accredited web pentester hired by companies to pentest their website again ?

Silly ponyfuk<3

Puller - August 15, 2014, 10:54 am

Really nice tut, something new and I learned something thanks :)

-Acid - August 15, 2014, 11:02 am

Yes..this is something great and marvelous thing i learned today.
thanks for this mate..
?

lenaos - August 15, 2014, 12:04 pm

Great tutorial man.

I learned something from this.

Skorp - August 15, 2014, 12:07 pm

Good thread my son, always HQ indeed.

ldkwntu - August 15, 2014, 2:36 pm

That's a nice tutorial, but it sounds like you think post-based XSS isn't dangerous.

In fact in most cases there is little difference between post-based and get-based XSS.

Th3PonyWizard - August 15, 2014, 2:57 pm

(Aug 15, 2014 02:36 PM)ldkwntu Wrote: [\[CITE\]](#) That's a nice tutorial, but it sounds like you think post-based XSS isn't dangerous.

In fact in most cases there is little difference between post-based and get-based XSS. Exploiting the most common uses of Reflected XSS (Cookie Stealing and Phishing) requires your XSS to be GET oriented so you'd easily persuade your target through social engineering to click it. POST based XSS has it's risk as well.

ldkwntu - August 15, 2014, 3:34 pm

(Aug 15, 2014 02:57 PM)Sonder Wrote: [\[CITE\]](#) Exploiting the most common uses of Reflected XSS (Cookie Stealing and Phishing) requires your XSS to be GET oriented so you'd easily persuade your target through social engineering to click it. POST based XSS has it's risk as well.

Sorry, but you're mistaken. You send someone a link to a [vulnerable website](#) if the XSS is GET based.

However if the XSS is POST based, you'd send someone a link to a [website controlled by the attacker](#), which would submit a form to execute the XSS payload on the vulnerable website.

Edward Blake - August 15, 2014, 4:44 pm

Interesting. Like you say this is more of a social engineered attack vector rather than a hardened code vector. It would be interesting to somehow try and get it to do remote code inclusion and actually write executable code to the page itself. That would probably be asking to much given the circumstance and what you are doing here.

Nice tut regardless.

Soulo - August 15, 2014, 6:23 pm

(Aug 15, 2014 03:34 PM)ldkwntu Wrote: [\[CITE\]](#) Sorry, but you're mistaken. You send someone a link to a [vulnerable website](#) if the XSS is GET based.

However if the XSS is POST based, you'd send someone a link to a [website controlled by the attacker](#), which would submit a form to execute the XSS payload on the vulnerable website.

^This, although I would use hidden forms regardless of whether the XSS vuln needs a POST or GET request.

The victim would just have to visit any site that the attacker can write HTML/JS to, be it the attacker's or one he gained access to. Theoretically, reflected XSS (POST or GET) can be exploited without any social engineering if the attacker is able to pwn a site that the target frequents.

Th3PonyWizard - August 16, 2014, 2:58 am

(Aug 15, 2014 03:34 PM)ldkwntu Wrote: [\[CITE\]](#) Sorry, but you're mistaken. You send someone a link to a [vulnerable website](#) if the XSS is GET based.

However if the XSS is POST based, you'd send someone a link to a [website controlled by the attacker](#), which would submit a form to execute the XSS payload on the vulnerable website. Thanks for the correction, never even thought about it.

i use linnucks haha - August 18, 2014, 9:46 pm

cute tutorial hunnybuns interesting read.

SIM-1073426 - August 19, 2014, 12:30 am

Nice and straight forward tutorial, makes sense. After 3 years of programming PHP I've never even used \$_REQUEST. I guess that's a good thing.

Zabuza? - August 19, 2014, 12:41 am

How interesting. Great job Sonder. I'm going to look into this some more.

tux1 - August 19, 2014, 1:04 pm

This is an interesting tutorial, thanks for sharing!

Sicarius - August 19, 2014, 4:25 pm

Awsome tut Sonder, very helpful as per usual :)

Frosteh - August 21, 2014, 12:44 am

Learned something new today, thanks.

Th3PonyWizard - August 26, 2014, 3:34 pm

tfw people downvote a tutorial. SORRY FOR CONTRIBUTING

(Aug 21, 2014 12:44 AM)Frosteh Wrote: [\[CITE\]](#) Learned something new today, thanks.Sure thing :)