

# Re: PHP Passing Variables Between Pages and Security

---

*Source:* <http://coding.derkeiler.com/Archive/PHP/comp.lang.php/2006-02/msg00738.html>

---

- *From:* Justin Koivisto <[justin@xxxxxxxx](mailto:justin@xxxxxxxx)>
  - *Date:* Fri, 10 Feb 2006 12:28:01 -0600
- 

Kevin D. wrote:

"Skeets" <[skillet3232@xxxxxxxx](mailto:skillet3232@xxxxxxxx)> wrote in message  
[news:1139509124.096351.108150@xx](mailto:news:1139509124.096351.108150@xx)

i'm passing session and hidden variables between pages. not to mention post values.

i'm a little concerned that someone with sufficient knowledge could spoof these vlaues and manipulate the program.

is this a valid concern? i'm thinking i can check the submitting page setting up something around the following the following code...

```
$base_name = basename($_SERVER['PHP_SELF']);
```

is this a good bet? is there a better way?

tia...

ps – posted this on php.general and, after 2 days w/o a response, realized that probably wasn't the best place to post it.

this is a very interesting thread and i'm learning a lot (of course some of it is over my head)... i'd like to clarify something, what exactly are we defending against?

in other words, i understand the concept of someone spoofing to hack my application... but what does this mean if my application is a basic content manager for a website? what are the true repercussions and possible worst-case scenarios that can take place?

Worst-case? Total server control. More realistic case: access to your data and code. If you store and kind of personal data (email addresses of customers perhaps), then access to the database could mean stealing

## Re: PHP Passing Variables Between Pages and Security

and/or deleting all that data.

Things like SQL injection and cross-site scripting can be found using things like form spoofing. By trusting any data from an outside source (your definitions may vary, but files, databases, rss feeds, user input, sessions, cookies, etc.), you may be open to attack – whether it's a known or unknown type.

Take an example that I had once encountered in an application I had taken over... An inexperienced php programmer created a form for uploading a Word document. It was then parsed using a command-line utility and the data was then saved in the database. The script didn't check if it was actually a Word document, and the command-line utility didn't either. By uploading the right file (a custom-made C binary for linux in this case), I was able to run a buffer overflow in the machine, start a new process which listened on a certain port, and connect to it. In this case, it was basically a telnet daemon with root privileges and no password... It was all very cool, but even more eye-opening. ;)

Just FYI – the reason this worked was because of specific versions of specific software (and linux kernel options), but it was just a demonstration as to why you must practice defense in depth for all programming (not just the web).

--

Justin Koivisto, ZCE – [justin@xxxxxxxx](mailto:justin@xxxxxxxx)  
<http://koivi.com>

.