Joomla! custom reports using “TinyButStrong”

A “get started” guide: Part 2 – staying secure

Summary

Joomla! Administrators familiar with the database and with SQL can use TinyButStrong (TBS) to present information from the database in web pages without needing to create a module or component. The first article in this series looked at the creation of simple tables, delivering the required information in a simple layout. This article looks at the security issues around using TBS and suggests how TBS might be used most securely. The concepts developed here are presented as a practical example in part 3 of this series.

Purpose

The TBS plugin for Joomla! allows access to the Joomla! database and therefore has security implications. It has various security related options that can be used to balance convenience and security. Some of these options are discussed in this note.

There are no references here to information on general website security or on Joomla! security as these would rapidly become outdated. For such information, current web searches are the best approach.

The risks

TBS allows direct access to the database containing all the information in a Joomla! web site. This essentially bypasses the normal security mechanisms. These are there to ensure that information designated as available only to specific groups is indeed only available to those groups. If you set up a way of accessing information using TBS, it is up to you to ensure that information does not become more widely available than it should be. In some jurisdictions, the information in the Joomla! database may be covered by data privacy legislation. Another class of information within the database is that defining the Joomla! site itself. Unauthorised access to this may allow others to hack your system. The design of TBS is such that simply installing it does not compromise the security of your site. The extent to which others can access the data is determined by paramters set by you, the administrator.

TBS security mechanisms

Plugin access

The plugin can be configured as usable by all, by restricted & special users or by special users only. In some situations it may be possible to use the plugin access configuration as the primary security control. If so, administration and system design become much easier. In practice, it is likely that the data presentation facilities provided by TBS will be used for publicly available data as well as for restricted data. When this is the case, the data access security features and associated system design must be used to maintain security.

Data access

The TBS plugin for Joomla! can be operated in three different ways:

- Direct MergeBlocks
- External scripts
- Internal scripts

Each has its use and its security implications. The security issues arise from access to the database, not from the presentation of data on the web site, hence [square bracket] sections related to TBS templates will not be discussed in this article. The approaches used have parameters that can be set in the TBS plugin. These are discussed in the following sections. In addition, the article number of every article using TBS must be specified in the plugin's parameters. This feature can be circumvented by specifying “*” as the article number. Setting the article number to “*” is not recommended if anyone other than the administrator has editor rights.

Direct MergeBlocks

Usage

Data is accessed using a TBS block of the form

{tbs}mergeblock=blockname; sql=SQL statements {/tbs}

This would appear in a Joomla! article. It is a convenient way to get started with TBS as it keeps the data extraction and the presentation templates in the same place, thus simplifying the development process.

Parameters

The parameter Direct MergeBlock has the following possible settings:

- Forbidden
- Only SELECT queries (default)
- All query types and stored procedures

Issues

Because any SELECT command can be issued from the Direct MergeBlock, anyone with the authority to change the article can change the SELECT command to access any data in the database. You therefore need to be very sure that no one with the ability to edit articles might have any reason to hunt through the database for information beyond their normal authority.

If the plugin's Direct MergeBlock parameter is set to “All query types and stored procedures”, someone able to edit the article can easily gain access as a super-administrator to your site. This a serious security risk.

The facility can be useful as a development tool on a private network behind a firewall.

Recommendations

1. It is most appropriate for a personal website where only one person maintains the site.
2. Never use Direct MergeBlocks with “All query types and stored procedures” on a publicly accessible web site.
3. Set the “Direct MergeBlock” parameter to “Forbidden” unless you are running a site with very few people authorised to edit articles and on which there is no information protected by data privacy laws (e.g. names and contact information).

4. Set the “Direct MergeBlock” parameter to “only SELECT queries” if you want to use TBS but you don't want to use External Scripts.

External Scripts

Usage

Data is accessed using a TBS block of the form

{tbs}script=myscript.php{/tbs}

The file “myscript.php” is a PHP language file including TBS elements. If given as a simple file name, it should be stored in the directory given in the “Script Location” parameter of the plugin.

Parameters

The External Scripts parameter has the following possible settings:

- Forbidden
- Only from the Script location (default)
- All external scripts

Issues

An external script can contain any valid PHP, hence there is no restriction on what can be done by the script. At first, this might seem to be a recipe for complete insecurity[1] but read on. The important feature of the “External Scripts” mechanism is the ability to specify the location in which the script files can be found. This is done using the “Only from Script location” parameter setting. By setting this to be a directory on your web server but outside the web root directory, you can ensure that only those with FTP access to your site are able to amend the script. Typically only the super-administrator group would have FTP access to files outside the web root, hence the super-administrator can create the scripts, ensuring that no sensitive information is available through them. Those with the the authority to edit documents can be shown the script files without compromising the database and they can use the data created by the script files in templates, only releasing the information deemed proper by the script's author.

By contrast, setting the parameter to “All external scripts” hands anyone with the authority to edit articles the keys to your site. The open_basedir parameter of the php.ini can be used to limit the risk but if used in its normal form where scripts must lie within the web-site, it must be extended to permit the use of scripts in the TBS script directory.

If you are not a PHP programmer, don't be put off: there is an example script file in the appendix. If you can manipulate SQL as required by TBS, you shouldn't have any problem with the trivial PHP required.

Recommendations

1. Create all script files in a directory outside the web root. If the open_basedir parameter is set in your php.ini file, you may need to amend it, e.g.

   ```
   open_basedir = /home/sites/[mysite]/public_html:/home/sites/
   [mysite]/tbsscripts
   ```

   It is not in the scope of this article to explain the open_basedir parameter.

2. Nominate the directory in (1) above as the location for script files.

1 If you don't understand why, you probably shouldn't be administering a Joomla! site

3. Keep the “External Scripts” parameter at its default value of “Only from the Script location”.
4. Ensure that no sensitive fields (e.g. “password”) are accessible through External Scripts.

Internal Scripts

Usage

Within the HTML include two separate sections:

{tbs}embedded{/tbs}

which tells TBS to look for an internal script, and

<!--TBS

PHP code

-->

which contains PHP code with TBS elements.

Parameters

1. Forbidden
2. Allowed

Issues

Internal scripts provide all the security risks of external scripts with none of the safeguards.

There may be circumstances where it is easier to develop a script using the “Internal script” approach before converting it to an external script. This should only be considered for a personal web server behind a firewall and never for a publicly accessible site.

Recommendations

1. Don't use them.
2. Set the “Internal Scripts” parameter to “Forbidden”.

Conclusion

For publicly accessible web sites, use only “External Scripts” and store these outside the web root. Set both the “Direct MergeBlock” and “Internal Script” parameters to “Forbidden”.

Acknowledgement

Without the author and maintainer of TBS, “skrol29”, none of this would be possible. It is clear that TBS is both very well designed and very well implemented. Thank you, skrol29 and your associates.

CheshireCat

1 Jul 2011

Appendix 1

Sample External Script

At its simplest, the external script file will contain:

```
<?php
$TBS->MergeBlock('blockname','mysql','SQL code');
?>
```

where “blockname” is the name of the block as used in the template [square bracket] sections, “mysql” is the database type (it might be mysqli in some sites) and “SQL code” is the SQL you wish to run against the Joomla! Database. Note that although the examples use the same name for the block and for the PHP file, there is no necessity to make the names the same.

The first example from article 1 used a Direct MergeBlock:

```
{tbs}mergeblock=content;sql=SELECT title, hits FROM jos_content
ORDER BY hits DESC LIMIT 0, 10{/tbs}
```

This could be replaced by:

```
{tbs}script=content1.php{/tbs}
```

where file *content1.php* contains the following:

```
<?php
$TBS->MergeBlock('content','mysql','SELECT title, hits FROM
jos_content ORDER BY hits DESC LIMIT 0, 10');
?>
```

As noted in the main part of this article, the file *content1.php* should be in the directory nominated in the “Script Location” parameter and this directory should lie outside the web root.