WordPress

CVE-2013-7279

Context

Cross-site scripting (XSS) vulnerability in views/video-management/preview_video.php in the S3 Video plugin before 0.983 for WordPress allows remote attackers to inject arbitrary web script or HTML via the base parameter.

Problem

S3 Video plugin for WordPress is vulnerable to cross-site scripting, caused by improper validation of user-supplied input by the preview_video.php script. A remote attacker could exploit this vulnerability using the base parameter in a specially-crafted URL to execute script in a victim's Web browser within the security context of the hosting Web site, once the URL is clicked. An attacker could use this vulnerability to steal the victim's cookie-based authentication credentials.

Solution

Fixed security issue and tested with Wordpress 3.8 Fixed XSS scripting vulnerability in video preview script

Codes

"Arbitrary code execution", "Cross-site scripting (XSS)", "Steal credentials", "Unsanitized data", "Inject Javascript"

CVE-2013-5963

Context

WordPress allows remote attackers to execute arbitrary code by uploading a file with an executable extension, then accessing it via a direct request to the file in wp-content/uploads/wpdb/.

Problem

#1.run the Firefox browser #2.Then Add-ons Live HTTP headers in Firefox Install >> #https://addons.mozilla.org/en-us/firefox/addon/live-http-headers/ #3.Now the run Add-ons Live HTTP headers #4.Then go to this page site/[path]/wp-content/plugins/simple-dropbox-upload-form/multi.php?&height=500&width=1000&TB_iframe=true #5.Click the Choose File button Then select a file [shell.jpg] #6.Then click on the Start upload button #7.Now using Live HTTP headers uploaded files to PHP change [shell.php] #8.Find your Shell site/wp-content/uploads/wpdb/shell.php

Solution

Removed multi.php

Codes

"File path traversal", "Lack of sandbox", "Unrestricted file upload"

CVE-2013-5098

Context

Download Monitor 'sort' Parameter Cross Site Scripting Vulnerability

Problem

The Download Monitor plugin for WordPress is prone to a cross-site scripting vulnerability because it fails to properly sanitize user-supplied input. An attacker may leverage this issue to execute arbitrary script code in the browser of an unsuspecting user in the context of the affected site. This can allow the attacker to steal cookie-based authentication credentials and to launch other attacks.

Solution

Sanitized the sort parameter, by invoking the function sanitize_text_field()

Codes

"Cross-site scripting (XSS)", "Steal credentials", "Unsanitized data", "Sanitizing data"

CVE-2013-4954

Context

WordPress Pie Register Plugin 'wp-login.php' Multiple Cross Site Scripting Vulnerabilities

Problem

Pie Register plugin for WordPress is prone to multiple cross-site scripting vulnerabilities. An attacker may leverage these issues to execute arbitrary script code in the browser of an unsuspecting user in the context of the affected site. This may allow the attacker to steal cookie-based authentication credentials and launch other attacks. Input is not sanitized before being output on the screen:

```
<?php echo $_POST[‘pass1’];?>
```

```
<?php echo $_POST[‘pass2’];?>
```

Solution

Escaping of HTML/Javascript using html_entity_decode:

```
<?php echo htmlspecialchars($_POST[‘pass1’]);?> <?php echo htmlspecialchars($_POST[‘pass2’]);?>
```

Codes

"Arbitrary code execution", "Cross-site scripting (XSS)", "Steal credentials", "Escape user-supplied data"

CVE-2013-3532

Context

WordPress Spider Video Player Plugin 'theme' Parameter SQL Injection Vulnerability

Problem

Spider Video Player plugin for WordPress is prone to an SQL-injection vulnerability because it fails to sufficiently sanitize user-supplied data before using it in an SQL query. Exploiting this issue could allow an attacker to compromise the application, access or modify data, or exploit latent vulnerabilities in the underlying database. Video Player Plugin for WordPress is vulnerable to SQL injection. A remote attacker could send specially-crafted SQL statements to the settings.php script using the playlist and theme parameters, which could allow the attacker to view, add, modify or delete information in the back-end database.

Solution

Sanitize input

Codes

"SQL injection", "Unrestricted access to database"

CVE-2013-3529

Context

WordPress FuneralPress Plugin Multiple HTML Injection Vulnerabilities

Problem

The FuneralPress plugin for WordPress is prone to multiple HTML-injection vulnerabilities because it fails to properly sanitize user-supplied input. Attacker-supplied HTML and script code would run in the context of the affected browser, potentially allowing the attacker to or control how the site is rendered to the user. Other attacks are also possible.

Solution

sanitize text fields

Codes

"Arbitrary code execution", "Steal data", "Steal credentials", "Unsanitized data", "HTML Injection"

CVE-2013-3526

Context

WordPress Traffic Analyzer Plugin 'aoid' **Parameter Cross Site Scripting Vulnerability**

Problem

The Traffic Analyzer plugin for WordPress is prone to a cross-site scripting vulnerability because it **fails to properly sanitize user-supplied input**. An attacker may leverage this issue to **execute arbitrary script code** in the browser of an unsuspecting user in the context of the affected site. This can allow the attacker to **steal cookie-based authentication credentials and launch other attacks.**

Solution

sanitize input

Codes

"Cross-site scripting (XSS)", "Escape user-supplied data"

CVE-2013-3491

Context

WordPress Sharebar Plugin CVE-2013-3491 **Cross Site Request Forgery Vulnerability**

Problem

The Sharebar plugin for WordPress is **prone to a cross-site request-forgery vulnerability**. Exploiting this issue may allow a remote attacker to **perform certain unauthorized actions** in the context of the affected application. Other attacks are also possible.

Solution

sanitize input

Codes

"Privilege elevation", "Cross-site request forgery (CSRF)", "Lack of sandbox"

CVE-2013-3262

Context

Download Monitor 'p' Parameter **Cross Site Scripting Vulnerability**

Problem

The Download Monitor plugin for WordPress is prone to a **cross-site scripting** vulnerability because it fails to **properly sanitize user-supplied input**. An attacker may leverage this issue to execute arbitrary script code in the browser of an unsuspecting user in the context of the affected site. This can allow the attacker to **steal cookie-based authentication credentials** and to launch other attacks.

Solution

Description: **Note: This plugin is no longer actively developed nor maintained! However, a rewrite is planned - http://mikejolley.com/2013/04/the-new-download-monitor-plugin/** Manage downloads on your site, view and show hits, and output in posts. sanitize_text_field to prevent XSS in admin

Codes

"Cross-site scripting (XSS)", "Steal credentials", "Unsanitized data"

CVE-2013-3253

Context

WordPress Xhanch - My Twitter Plugin CVE-2013-3253 Cross Site Request Forgery Vulnerability

Problem

The Xhanch - My Twitter plugin for WordPress is prone to a cross-site request-forgery vulnerability. Exploiting this issue may allow a remote attacker to perform certain unauthorized actions in the context of the affected application. Other attacks are also possible.

Solution

Security n hashtag

Codes

"Cross-site request forgery (CSRF)"

CVE-2013-2741

Context

The final step in the importbuddy backup restoration process is supposed to remove importbuddy.php from the root of the site, however this step often fails (most commonly as a result of filesystem permissions) allowing an attacker access to some or all of the functions and information provided by importbuddy.php. An access password for importbuddy does not appear to be a mandatory requirement.

Problem

The name of the backup file contains a random string intended to prevent an attacker from guessing its value. However if backup files are present, browsing to http://site/importbuddy.phpwill expose their filenames; these can then be used to download the files from the site: backup-zipfile-date-randomstring.zip The desired backup file can be retrieved with: wget http://site/backup-zipfile-date-randomstring.zip The backup consists of a zip archive containing the wordpress directory, complete with wp-config.php and often a .sql dump containing a full copy of the wordpress database and any other databases the backupbuddy plugin has been configured to include. Importbuddy also presents the option to upload a backup on step 1 of the restoration process, potentially allowing defacement or deletion and also trojanning the site if an existing backup is available. Additionally there are issues affecting the 'step' query string field. This has a differing impact depending on the version of Backupbuddy targeted: http://site/importbuddy.php?step=1

Solution

Forcing the user to set a password (and fixing the authentication bypass) would go some way to mitigating the risk of importbuddy.php not being deleted.

Codes

"Overwrite files"

CVE-2013-2640

Context

WordPress does not properly restrict access to unspecified Ajax functions, which allows remote attackers to modify plugin settings and conduct cross-site scripting (XSS) attacks via unspecified vectors related to "formData=save" requests

Problem

WordPress does not properly restrict access to unspecified Ajax functions, which allows remote attackers to modify plugin settings and conduct cross-site scripting (XSS) attacks via unspecified vectors related to "formData=save" requests

Solution

Security vulnerability has been patched with is_user_logged_in() and now submitted to first review

Codes

"Cross-site scripting (XSS)", "Unsanitized data", "Not enforcing private code"

CVE-2013-2501

Context

WordPress Terillion Reviews Plugin Profile Id HTML Injection Vulnerability

Problem

The Terillion Reviews plugin for WordPress is prone to an HTML-injection vulnerability because it fails to properly sanitize user-supplied input before using it in dynamically generated content. Successful exploits will allow attacker-supplied HTML and script code to run in the context of the affected browser, potentially allowing the attacker to steal cookie-based authentication credentials or to control how the site is rendered to the user. Other attacks are also possible.

Solution

Fix security vulnerability by sanitizing user input

Codes

"Cross-site scripting (XSS)", "Lack of sandbox"

CVE-2013-2204

Context

Content Spoofing in the MoxieCode (TinyMCE) MoxiePlayer project

Problem

Flash doesn't recognize '#' symbol as the beginning of the fragment to ignore, so if '?' mark follows, remaining part of the url will still be interpreted as application parameters...

Solution

Consider part of url after '?' as querystring, no matter what preceeds it.

Codes

"Lack of sandbox", "Content spoofing"

CVE-2013-1464

Context

WordPress Audio Player SWF Cross Site Scripting

Problem

WordPress Audio Player Plugin 'playerID' Parameter Cross Site Scripting Vulnerability The Audio Player plugin for WordPress is prone to a cross-site scripting vulnerability because it fails to properly sanitize user-supplied input. An attacker may leverage this issue to execute arbitrary script code in the browser of an unsuspecting user in the context of the affected site. This can allow the attacker to steal cookie-based authentication credentials and launch other attacks.

Solution

sanitize input

Codes

"Cross-site scripting (XSS)", "HTML Injection", "Inject Javascript"

CVE-2013-1409

Context

Cross-Site Scripting (XSS) in CommentLuv wordpress plugin

Problem

The vulnerability exists due to **insufficient filtration of user-supplied data** in "_ajax_nonce" HTTP POST parameter in the "/wp-admin/admin-ajax.php" script. A remote attacker can trick a logged-in administrator to open a specially crafted link and **execute arbitrary HTML and script code in browser** in context of the vulnerable website. PoC (Proof-of-Concept) below uses the "alert()" JavaScript function to display administrator's cookiees

Solution

Upgrade to CommentLuv 2.92.4

filter user supplied data

Codes

"Cross-site scripting (XSS)", "Unsanitized data", "Lack of sandbox"

CVE-2013-0735

Context

WordPress Mingle Forum Plugin Multiple **SQL Injection and Cross Site Scripting Vulnerabilities**

Problem

The Mingle Forum plug-in for WordPress is prone to **multiple SQL-injection and cross-site scripting vulnerabilities** because it f**ails to sufficiently sanitize user-supplied input.** Exploiting these vulnerabilities could allow an attacker to **steal cookie-based authentication credentials, compromise the application, access or modify data, or exploit latent vulnerabilities in the underlying database.**

Solution

sanitize user-supplied data

Codes

"SQL injection", "Unrestricted access to database"

CVE-2013-0734

Context

WordPress Mingle Forum Plugin Multiple **SQL Injection and Cross Site Scripting Vulnerabilities**

Problem

The Mingle Forum plug-in for WordPress is prone to **multiple SQL-injection and cross-site scripting vulnerabilities** because it **fails to sufficiently sanitize user-supplied input**. Exploiting these vulnerabilities could allow an attacker to **steal cookie-based authentication credentials, compromise the application, access or modify data, or exploit latent vulnerabilities in the underlying database.**

Solution

sanitize user-supplied data

Codes

"SQL injection", "Unrestricted access to database"

CVE-2013-0731

Context

WordPress MailUp Plugin **Security Bypass Vulnerability**

Problem

WordPress **does not properly restrict access** to unspecified Ajax functions, which **allows remote attackers to modify plugin settings and conduct cross-site scripting (XSS)** attacks by setting the wordpress_logged_in cookie.

Solution

Security vulnerability has been patched with is_user_logged_in() and now submitted to first review

Codes

"Cross-site scripting (XSS)", "Unsanitized data", "Not enforcing private code", "Bypass protection mechanism"

CVE-2012-6527

Context

Cross-site scripting (XSS) vulnerability in the My Calendar plugin before 1.10.2 for WordPress allows remote attackers to inject arbitrary web script or HTML via the PATH_INFO.

Problem

My Calendar plugin for WordPress is vulnerable to cross-site scripting, caused by improper validation of user-supplied input. A remote attacker could exploit this vulnerability using a specially-crafted URL to execute script in a victim's Web browser within the security context of the hosting Web site, once the URL is clicked. An attacker could use this vulnerability to steal the victim's cookie-based authentication credentials.

Solution

escape urls before use. Confirm that event ids are integers.

Codes

"Cross-site scripting (XSS)", "Inject Javascript", "Sanitize data (enforce expected datatype)", "User-assisted attack"

CVE-2012-5328

Context

Multiple SQL injection vulnerabilities in the Mingle Forum plugin 1.0.32.1 and other versions before 1.0.33 for WordPress might allow remote authenticated users to execute arbitrary SQL commands via the (1) memberid or (2) groupid parameters in a removemember action or (3) id parameter to fs-admin/fs-admin.php, or (4) edit_forum_id parameter in an edit_save_forum action to fs-admin/wpf-edit-forum-group.php.

Problem

Sql injection vulnerabilities.

Solution

escape data in post request before performing sql operations.

Codes

"SQL injection", "Unrestricted access to database"

CVE-2012-5327

Context

Multiple SQL injection vulnerabilities in fs-admin/fs-admin.php in the Mingle Forum plugin 1.0.32.1 and other versions before 1.0.33 for WordPress allow remote authenticated users to execute arbitrary SQL commands via the (1) delete_usrgrp[] parameter in a delete_usergroups action, (2) usergroup parameter in an add_user_togroup action, or (3) add_forum_group_id parameter in an add_forum_submit action.

Problem

WordPress Mingle Forum Plugin is vulnerable to SQL injection. A remote attacker could send specially-crafted SQL statements to the admin.php script using the multiple parameters, which could allow the attacker to view, add, modify or delete information in the back-end database.

Solution

escape data in post requests before doing database operations.

Codes

"Escape user-supplied data", "SQL injection"

CVE-2012-4283

Context

Cross-site scripting (XSS) vulnerability in the Login With Ajax plugin before 3.0.4.1 for WordPress allows remote attackers to inject arbitrary web script or HTML via the callback parameter.

Problem

XSS in Login with Ajax plugin

Solution

use regex to match the expected query string

Codes

"Cross-site scripting (XSS)", "Unsanitized data", "HTML Injection", "Inject Javascript"

CVE-2012-4273

Context

Cross-site scripting (XSS) vulnerability in libs/xing.php in the 2 Click Social Media Buttons plugin before 0.34 for WordPress allows remote attackers to inject arbitrary web script or HTML via the xing-url parameter.

Problem

2 Click Social Media Buttons plugin for WordPress is vulnerable to cross-site scripting, caused by improper validation of user-supplied input by the xing.php script. A remote attacker could exploit this vulnerability using the xing-url parameter in a specially-crafted URL to execute script in a victim's Web browser within the security context of the hosting Web site, once the URL is clicked. An attacker could use this vulnerability to steal the victim's cookie-based authentication credentials.

Solution

strip tags from url

Codes

"Perform security check on unsanitized data", "Cross-site scripting (XSS)"

CVE-2012-4272

Context

Multiple cross-site scripting (XSS) vulnerabilities in the 2 Click Social Media Buttons plugin before 0.34 for WordPress allow remote attackers to inject arbitrary web script or HTML via unspecified vectors related to the "processing of the buttons of Xing and Pinterest".

Problem

XSS vulnerabilities in Click Social Media Buttons plugin

Solution

strip tags before decoding raw url

Codes

"Perform security check on unsanitized data", "Cross-site scripting (XSS)", "Unsanitized data"

CVE-2012-4271

Context

Multiple cross-site scripting (XSS) vulnerabilities in bad-behavior-wordpress-admin.php in the Bad Behavior plugin before 2.0.47 and 2.2.x before 2.2.5 for WordPress allow remote attackers to inject arbitrary web script or HTML via the (1) PATH_INFO, (2) httpbl_key, (3) httpbl_maxage, (4) httpbl_threat, (5) reverse_proxy_addresses, or (6) reverse_proxy_header parameter.

Problem

Bad Behavior plugin for WordPress is vulnerable to cross-site scripting, caused by improper validation of user-supplied input by the options-general.php script. A remote attacker could exploit this vulnerability using multiple parameters in a specially-crafted URL to execute script in a victim's Web browser within the security context of the hosting Web site, once the URL is clicked. An attacker could use this vulnerability to steal the victim's cookie-based authentication credentials.

Solution

escape url before use (sanitize)

Codes

"Cross-site scripting (XSS)", "Unsanitized data", "Escape user-supplied data"

CVE-2012-4268

Context

Cross-site scripting (XSS) vulnerability in bulletproof-security/admin/options.php in the BulletProof Security plugin before .47.1 for WordPress allows remote attackers to inject arbitrary web script or HTML via the HTTP_ACCEPT_ENCODING header.

Problem

BulletProof Security plugin for WordPress is vulnerable to cross-site scripting, caused by improper validation of user-supplied input by the admin.php script. A remote attacker could exploit this vulnerability using the page parameter in a specially-crafted URL to execute script in a victim's Web browser within the security context of the hosting Web site, once the URL is clicked. An attacker could use this vulnerability to steal the victim's cookie-based authentication credentials.

Solution

filter and sanitize input string before use.

Codes

"Steal data", "Cross-site scripting (XSS)"

CVE-2012-4264

Context

Multiple cross-site scripting (XSS) vulnerabilities in the Better WP Security (better_wp_security) plugin before 3.2.5 for WordPress allow remote attackers to inject arbitrary web script or HTML via unspecified vectors related to "server variables," a different vulnerability than CVE-2012-4263.

Problem

XSS vulnerabilities in Better WP Security plugin

Solution

filter and sanitize string before use.

Codes

"Arbitrary code execution", "Cross-site scripting (XSS)", "Unsanitized data"

CVE-2012-4263

Context

Cross-site scripting (XSS) vulnerability in inc/admin/content.php in the Better WP Security (better_wp_security) plugin before 3.2.5 for WordPress allows remote attackers to inject arbitrary web script or HTML via the HTTP_USER_AGENT header.

Problem

The value of the User-Agent HTTP header is copied into the HTML document as plain text between tags. The payload a712378089b4648b was submitted in the User-Agent HTTP header. This input was echoed unmodified in the application's response. This proof-of-concept attack demonstrates that it is possible to inject arbitrary JavaScript into the application's response. Issue background Reflected cross-site scripting vulnerabilities arise when data is copied from a request and echoed into the application's immediate response in an unsafe way. An attacker can use the vulnerability to construct a request which, if issued by another application user, will cause JavaScript code supplied by the attacker to execute within the user's browser in the context of that user's session with the application. The attacker-supplied code can perform a wide variety of actions, such as stealing the victim's session token or login credentials, performing arbitrary actions on the victim's behalf, and logging their keystrokes. Users can be induced to issue the attacker's crafted request in various ways. For example, the attacker can send a victim a link containing a malicious URL in an email or instant message. They can submit the link to popular web sites that allow content authoring, for example in blog comments. And they can create an innocuous looking web site which causes anyone viewing it to make arbitrary cross-domain requests to the vulnerable application (using either the GET or the POST method). The security impact of cross-site scripting vulnerabilities is dependent upon the nature of the vulnerable application, the kinds of data and functionality which it contains, and the other applications which belong to the same domain and organisation. If the application is used only to display non-sensitive public content, with no authentication or access control functionality, then a cross-site scripting flaw may be considered low risk. However, if the same application resides on a domain which can access cookies for other more security-critical applications, then the vulnerability could be used to attack those other applications, and so may be considered high risk. Similarly, if the organisation which owns the application is a likely target for phishing attacks, then the vulnerability could be leveraged to lend credibility to such attacks, by injecting Trojan functionality into the vulnerable application, and exploiting users' trust in the organisation in order to capture credentials for other applications which it owns. In many kinds of application, such as those providing online banking functionality, cross-site scripting should always be considered high risk. Issue remediation In most situations where user-controllable data is copied into application responses, cross-site scripting attacks can be prevented using two layers of defences: Input should be validated as strictly as possible on arrival, given the kind of content which it is expected to contain.

For example, personal names should consist of alphabetical and a small range of typographical characters, and be relatively short; a year of birth should consist of exactly four numerals; email addresses should match a well-defined regular expression. Input which fails the validation should be rejected, not sanitised. User input should be HTML-encoded at any point where it is copied into application responses. All HTML metacharacters, including < > " ' and =, should be replaced with the corresponding HTML entities (< > etc). In cases where the application's functionality allows users to author content using a restricted subset of HTML tags and attributes (for example, blog comments which allow limited formatting and linking), it is necessary to parse the supplied HTML to validate that it does not use any dangerous syntax; this is a non-trivial task.

Exploit Example:

```
Request
GET /wp-admin/admin.php?page=better_wp_security HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:11.0)
Gecko/20100101 Firefox/11.0a7123<script>alert(1)</script>78089b4648b
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
Proxy-Connection: keep-alive
Referer: http://127.0.0.1/wp-admin/options-general.php
Cookie:
wordpress_5c016e8f0f95f039102cbe8366c5c7f3=admin%7C1333587813%7C73fe3e4d525d2460588947c4b7a03114;
wp-settings-1=widgets_access%3Doff%26uploader%3D1;
wp-settings-time-1=1333368822; wordpress_test_cookie=WP+Cookie+check;
wordpress_logged_in_5c016e8f0f95f039102cbe8366c5c7f3=admin%7C1333587813%7C7b961d6e5caea2c784f282c5ed426964;
bb2_screener_=1333415711+127.0.0.1; PHPSESSID=j3l493obmauq7akebg8g3jb4k3


Response
HTTP/1.1 200 OK
```

```
Date: Tue, 03 Apr 2012 02:05:00 GMT
Server: Apache/2.2.20 (Ubuntu)
X-Powered-By: PHP/5.3.6-13ubuntu3.6
Expires: Wed, 11 Jan 1984 05:00:00 GMT
Last-Modified: Tue, 03 Apr 2012 02:05:00 GMT
Cache-Control: no-cache, must-revalidate, max-age=0
Pragma: no-cache
X-Frame-Options: SAMEORIGIN
Vary: Accept-Encoding
Content-Length: 35849
Content-Type: text/html; charset=UTF-8



<!DOCTYPE html>
<!--[if IE 8]>
<html xmlns="http://www.w3.org/1999/xhtml" class="ie8" dir="ltr" lang="en-US">
<![endif]-->
<!--[if !(IE 8) ]><!-->
<html xmlns="http://www.w3.org/1999/xhtml" dir="ltr
...[SNIP]...
<strong>Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:11.0)
Gecko/20100101
Firefox/11.0a7123<script>alert(1)</script>[Injected Script]78089b4648b</strong>
```

Solution

Better Sanitization to eliminate possible XSS attacks

Codes

"Arbitrary code execution", "Cross-site scripting (XSS)", "Sanitize data (enforce expected datatype)"

CVE-2012-3576

Context

Unrestricted file upload vulnerability in php/upload.php in the wpStoreCart plugin before 2.5.30 for WordPress allows remote attackers to execute arbitrary code by uploading a file with an executable extension, then accessing it via a direct request to the file in uploads/wpstorecart.

Problem

wpStoreCart plugin for WordPress could allow a remote attacker to upload arbitrary files, caused by the improper validation of file extensions by the upload.php script. By sending a specially-crafted HTTP request, a remote attacker could exploit this vulnerability to upload a malicious PHP script, which could allow the attacker to execute arbitrary PHP code on the vulnerable system.

Solution

validate file extensions

Codes

"Lack of sandbox", "Unrestricted file upload"

CVE-2012-2920

Context

Cross-site scripting (XSS) vulnerability in the userphoto_options_page function in user-photo.php in the User Photo plugin before 0.9.5.2 for WordPress allows remote attackers to inject arbitrary web script or HTML via the PATH_INFO to wp-admin/options-general.php. NOTE: some of these details are obtained from third party information.

Problem

User Photo plugin for WordPress is vulnerable to cross-site scripting, caused by improper validation of user-supplied input by the options-general.php script. A remote attacker could exploit this vulnerability using the '$_SERVER['REQUEST_URI']' parameter in a specially-crafted URL to execute script in a victim's Web browser within the security context of the hosting Web site, once the URL is clicked. An attacker could use this vulnerability to steal the victim's cookie-based authentication credentials.

Solution

escape url before use

Codes

"Unsanitized parameter", "Cross-site scripting (XSS)", "Escape user-supplied data"

CVE-2012-2916

Context

Cross-site scripting (XSS) vulnerability in sabre_class_admin.php in the SABRE plugin before 2.1 for WordPress allows remote attackers to inject arbitrary web script or HTML via the active_option parameter to wp-admin/tools.php.

Problem

SABRE plugin for WordPress is vulnerable to cross-site scripting, caused by improper validation of user-supplied input by the tools.php script. A remote attacker could exploit this vulnerability using the active_option parameter in a specially-crafted URL to execute script in a victim's Web browser within the security context of the hosting Web site, once the URL is clicked. An attacker could use this vulnerability to steal the victim's cookie-based authentication credentials.

Solution

sanitize text before using it.

Codes

"Unsanitized parameter", "Cross-site scripting (XSS)", "Inject Javascript", "Not escaping characters"

CVE-2012-2759

Context

Cross-site scripting (XSS) vulnerability in login-with-ajax.php in the Login With Ajax (aka login-with-ajax) plugin before 3.0.4.1 for WordPress allows remote attackers to inject arbitrary web script or HTML via the callback parameter in a lostpassword action to wp-login.php.

Problem

Login With Ajax plugin for WordPress is vulnerable to cross-site scripting, caused by improper validation of user-supplied input by the login-with-ajax.php script. A remote attacker could exploit this vulnerability using the JSON callback parameter in a specially-crafted URL to execute script in a victim's Web browser within the security context of the hosting Web site, once the URL is clicked. An attacker could use this vulnerability to steal the victim's cookie-based authentication credentials.

Solution

Sanitize data before use

Codes

"Unsanitized parameter", "Cross-site scripting (XSS)", "Sanitize data (enforce expected datatype)"

CVE-2012-2402

Context

wp-admin/plugins.php in WordPress before 3.3.2 allows remote authenticated site administrators to bypass intended access restrictions and deactivate network-wide plugins via unspecified vectors.

Problem

WordPress could allow a remote attacker to bypass security restrictions, caused by an error in plugins.php when handling network wide plugins. A remote attacker could exploit this vulnerability to bypass security restrictions and gain unauthorized administrative access to the vulnerable application.

Solution

check that the authenticated user is a wp admin before allowing them to disable plugins

Codes

CVE-2012-1785

Context

kg_callffmpeg.php in the Video Embed & Thumbnail Generator plugin before 2.0 for WordPress allows remote attackers to execute arbitrary commands via unspecified vectors.

Problem

Video Embed & Thumbnail Generator plugin for WordPress could allow a remote attacker to execute arbitrary codeon the system, caused by the improper validation of user-supplied input in exec() function by the kg_callffmpeg.php script. By persuading a victim to visit a specially-crafted Web page, a remote attacker could exploit this vulnerability to inject and execute arbitrary shell code on the vulnerable system.

Solution

validate input before using

Codes

"Steal data", "Cross-site scripting (XSS)", "improper validation of user-supplied input"

CVE-2012-1205

Context

PHP remote file inclusion vulnerability in relocate-upload.php in Relocate Upload plugin before 0.20 for WordPress allows remote attackers to execute arbitrary PHP code via a URL in the abspath parameter.

Problem

Allowed anybody in include remote files

Solution

Adopted proper 'wp_ajax_' action, to close off a major security issue. only admin should have access to certain actions ('wp_ajax_relocate_upload', 'relocate_upload_js_action')

Codes

CVE-2012-1125

Context

Unrestricted file upload vulnerability in uploadify/scripts/uploadify.php in the Kish Guest Posting plugin before 1.2 for WordPress allows remote attackers to execute arbitrary code by uploading a file with a PHP extension, then accessing it via a direct request to the file in the directory specified by the folder parameter.

Problem

Kish Guest Posting plugin for WordPress could allow a remote attacker to upload arbitrary files, caused by the improper validation of file extensions by the uploadify.php script. By sending a direct request using the folder parameter, a remote attacker could exploit this vulnerability to upload a malicious PHP script, which could allow the attacker to execute arbitrary PHP code on the vulnerable system.

Solution

Check the extension of the file to make sure its an allowable type. Do not allow folder creation.

Codes

"Unsanitized data", "Lack of sandbox", "Unrestricted file upload"

CVE-2012-1068

Context

Cross-site scripting (XSS) vulnerability in the rc_ajax function in core.php in the WP-RecentComments plugin before 2.0.7 for WordPress allows remote attackers to inject arbitrary web script or HTML via the page parameter, related to AJAX paging.

Problem

WP-RecentComments Plugin for WordPress is vulnerable to cross-site scripting, caused by improper validation of user-supplied input by the core.php script. A remote attacker could exploit this vulnerability using the page parameter in a specially-crafted URL to execute script in a victim's Web browser within the security context of the hosting Web site, once the URL is clicked. An attacker could use this vulnerability to steal the victim's cookie-based authentication credentials.

Solution

sanitize input (make sure page is an int) before using the data.

Codes

"Cross-site scripting (XSS)", "Unsanitized data", "improper validation of user-supplied input", "Sanitize data (enforce expected datatype)"

CVE-2012-1011

Context

actions.php in the AllWebMenus plugin 1.1.8 for WordPress allows remote attackers to bypass intended access restrictions to upload and execute arbitrary PHP code by setting the HTTP_REFERER to a certain value, then uploading a ZIP file containing a PHP file, then accessing it via a direct request to the file in an unspecified directory.

Problem

WordPress AllWebMenus Plugin could allow a remote attacker to upload arbitrary files, caused by the improper validation of file extensions by the actions.php script. By sending a specially-crafted HTTP request, a remote attacker could exploit this vulnerability to upload a malicious PHP script, which could allow the attacker to execute arbitrary PHP code on the vulnerable system. lack of checks in script actions.php allowed malicious user to upload any file to the vulnerable server. Create a file (For example, Wordpress_security.php, with this content: `echo <php echo '6Scan to the rescue';  ?>`. Compress it with zip to awm.zip

Solution

checks the source referrer

Codes

"Unrestricted file upload"

CVE-2012-1010

Context

Unrestricted file upload vulnerability in actions.php in the AllWebMenus plugin before 1.1.8 for WordPress allows remote attackers to execute arbitrary PHP code by uploading a ZIP file containing a PHP file, then accessing it via a direct request to the file in an unspecified directory.

Problem

WordPress AllWebMenus Plugin could allow a remote attacker to upload arbitrary files, caused by the improper validation of file extensions by the actions.php script. By sending a specially-crafted HTTP request, a remote attacker could exploit this vulnerability to

upload a malicious PHP script, which could allow the attacker to execute arbitrary PHP code on the vulnerable system. lack of checks in script actions.php allowed malicious user to upload any file to the vulnerable server.

Solution

checks the source referrer,

Codes

"Lack of sandbox", "Unrestricted file upload"

CVE-2012-0934

Context

PHP remote file inclusion vulnerability in ajax/savetag.php in the Theme Tuner plugin for WordPress before 0.8 allows remote attackers to execute arbitrary PHP code via a URL in the tt-abspath parameter.

Problem

WordPress Theme Tuner Plugin could allow a remote attacker to include arbitrary files. A remote attacker could send a specially-crafted URL request to the savetag.php script using the tt-abspath parameter to specify a malicious file from a remote system, which could allow the attacker to execute arbitrary code on the vulnerable Web server.

Solution

sanitize data before using it

Codes

"Remote Code Execution", "File path traversal", "Remote file inclusion"

CVE-2011-5264

Context

Cross-site scripting (XSS) vulnerability in lazyest-backup.php in the Lazyest Backup plugin before 0.2.2 for WordPress allows remote attackers to inject arbitrary web script or HTML via the xml_or_all parameter

Problem

Lazyest Backup Plugin for WordPress is vulnerable to cross-site scripting, caused by improper validation of user-supplied input. A remote attacker could exploit this vulnerability using the xml_or_all parameter in a specially-crafted URL to execute script in a victim's Web browser within the security context of the hosting Web site, once the URL is clicked. An attacker could use this vulnerability to steal the victim's cookie-based authentication credentials.

Solution

sanitize input before using it

Codes

"Steal data", "Perform security check on unsanitized data", "Unsanitized parameter", "Cross-site scripting (XSS)"

CVE-2011-5226

Context

Cross-site request forgery (CSRF) vulnerability in wordpress_sentinel.php in the Sentinel plugin 1.0.0 for WordPress allows remote attackers to hijack the authentication of an administrator for requests that trigger snapshots.

Problem

Sentinel Plugin for WordPress is vulnerable to cross-site request forgery, caused by improper validation of user-supplied input. By persuading an authenticated user to visit a malicious Web site, a remote attacker could send a malformed HTTP request to perform

unauthorized actions. An attacker could exploit this vulnerability to perform cross-site scripting attacks, Web cache poisoning, and other malicious activities.

Solution

escape html before using the input (sanitize)

Codes

"Unsanitized data", "Cross-site request forgery (CSRF)", "Lack of sandbox"

CVE-2011-5225

Context

Cross-site scripting (XSS) vulnerability in wordpress_sentinel.php in the Sentinel plugin 1.0.0 for WordPress allows remote attackers to inject arbitrary web script or HTML via unknown vectors.

Problem

Sentinel Plugin for WordPress is vulnerable to cross-site scripting, caused by improper validation of user-supplied input. A remote attacker could exploit this vulnerability using a specially-crafted URL to execute script in a victim's Web browser within the security context of the hosting Web site, once the URL is clicked. An attacker could use this vulnerability to steal the victim's cookie-based authentication credentials.

Solution

sanitize input before use

Codes

"Perform security check on unsanitized data", "Unsanitized parameter", "Cross-site scripting (XSS)"

CVE-2011-5224

Context

SQL injection vulnerability in the Sentinel plugin 1.0.0 for WordPress allows remote attackers to execute arbitrary SQL commands via unspecified vectors.

Problem

Sentinel Plugin for WordPress is vulnerable to SQL injection. A remote attacker could send specially-crafted SQL statements which could allow the attacker to view, add, modify or delete information in the back-end database.

Solution

validate data before using it for sql operations

Codes

"SQL injection", "Unrestricted access to database"

CVE-2011-5216

Context

SQL injection vulnerability in ajax.php in SCORM Cloud For WordPress plugin before 1.0.7 for WordPress allows remote attackers to execute arbitrary SQL commands via the active parameter

Problem

SCORM Cloud for WordPress is vulnerable to SQL injection. A remote attacker could send specially-crafted SQL statements to the ajax.php script using the active parameter, which could allow the attacker to view, add, modify or delete information in the back-end database.

Solution

validate data before performing sql operations

Codes

"SQL injection", "Unrestricted access to database"

CVE-2011-5194

Context

Cross-site scripting (XSS) vulnerability in vendors/samswhois/samswhois.inc.php in the Whois Search plugin before 1.4.2.3 for WordPress allows remote attackers to inject arbitrary web script or HTML via the domain parameter, a different vulnerability than CVE-2011-5193.

Problem

The domain parameter allows for XSS

Solution

Validate input around the domain parameter

Codes

"Cross-site scripting (XSS)", "Unsanitized data"

CVE-2011-5192

Context

Cross-site scripting (XSS) vulnerability in pretty-bar.php in Pretty Link Lite plugin before 1.5.6 for WordPress allows remote attackers to inject arbitrary web script or HTML via the slug parameter, a different vulnerability than CVE-2011-5191.

Problem

Characters are not escaped around the slug parameter and allows for XSS.

Solution

Fixed a cross-site scripting vulnerability that could have affected a very small number of users Fix XSS near the slug parameter

Codes

"Cross-site scripting (XSS)", "Unsanitized data", "Escape user-supplied data"

CVE-2011-5191

Context

Cross-site scripting (XSS) vulnerability in pretty-bar.php in Pretty Link Lite plugin before 1.5.4 for WordPress allows remote attackers to inject arbitrary web script or HTML via the slug parameter, a different vulnerability than CVE-2011-5192.

Problem

Pretty link lite plugin allows XSS through via the slug parameter

Solution

Fixed an issue with Pretty Link Export link for Pro users check input to make sure XSS is not possible

Codes

"Perform security check on unsanitized data", "Cross-site scripting (XSS)", "Unsanitized data"

CVE-2011-5181

Context

Cross-site scripting (XSS) vulnerability in clickdesk.php in ClickDesk Live Support - Live Chat plugin 2.0 for WordPress allows remote attackers to inject arbitrary web script or HTML via the cdwidgetid parameter. NOTE: some of these details are obtained from third party information.

Problem

Based on exploit: https://www.exploit-db.com/exploits/36338/ ClickDesk Live Support plugin for WordPress is prone to a cross-site-scripting vulnerability because it fails to properly sanitize user-supplied input. An attacker may leverage this issue to execute arbitrary script code in the browser of an unsuspecting user in the context of the affected site. This can allow the attacker to steal cookie-based authentication credentials and launch other attacks. http://www.example.com/[path]/wp-content/plugins/clickdesk-live-support-chat/clickdesk.php?cdwidgetid=[xss]

Solution

Not found

Codes

"Cross-site scripting (XSS)", "Unsanitized data"

CVE-2011-5180

Context

Cross-site scripting (XSS) vulnerability in wp-1pluginjquery.php in the ZooEffect plugin 1.01 for WordPress allows remote attackers to inject arbitrary web script or HTML via the page parameter. NOTE: some of these details are obtained from third party information. NOTE: this has been disputed by a third party.

Problem

Based on exploit: https://www.exploit-db.com/exploits/36323/

1-jquery-photo-gallery-slideshow-flash plug-in for WordPress is prone to a cross-site-scripting vulnerability because it fails to sufficiently sanitize user-supplied data. An attacker may leverage this issue to execute arbitrary script code in the browser of an unsuspecting user in the context of the affected site. This can allow the attacker to steal cookie-based authentication credentials and launch other attacks. http://www.example.com/[path]/wp-content/plugins/1-jquery-photo-gallery-slideshow-flash/wp-1pluginjquery.php?page=[xss]

Solution

Not found

Codes

"Cross-site scripting (XSS)", "Unsanitized data"

CVE-2011-5128

Context

Multiple cross-site scripting (XSS) vulnerabilities in the Adminimize plugin before 1.7.22 for WordPress allow remote attackers to inject arbitrary web script or HTML via the page parameter to (1) inc-options/deinstall_options.php, (2) inc-options/theme_options.php, or (3) inc-options/im_export_options.php, or the (4) post or (5) post_ID parameters to adminimize.php, different vectors than CVE-2011-4926.

Problem

Adminimize plugin does not escape html attributes in several parameters which enables users to perform XSS

Solution

Escape html attributes in user input

Codes

"Cross-site scripting (XSS)", "Unsanitized data", "Not escaping characters"

CVE-2011-5107

Context

Cross-site scripting (XSS) vulnerability in post_alert.php in Alert Before Your Post plugin, possibly 0.1.1 and earlier, for WordPress allows remote attackers to inject arbitrary web script or HTML via the name parameter.

Problem

(Based on exploit)

"Alert Before Your Post" plugin for WordPress is prone to a cross-site scripting vulnerability because it fails to properly sanitize user-supplied input. An attacker may leverage this issue to execute arbitrary script code in the browser of an unsuspecting user in the context of the affected site. This can allow the attacker to steal cookie-based authentication credentials and launch other attacks.
http://www.example.com/[path]/wp-content/plugins/alert-before-your-post/trunk/post_alert.php?name=[xss]

Solution

Properly Sanitize Input

Codes

"Arbitrary code execution", "Steal data", "Unsanitized parameter", "Cross-site scripting (XSS)", "Steal credentials", "Unsanitized data"

CVE-2011-5106

Context

Cross-site scripting (XSS) vulnerability in edit-post.php in the Flexible Custom Post Type plugin before 0.1.7 for WordPress allows remote attackers to inject arbitrary web script or HTML via the id parameter.

Problem

When querying the id of the post, the Flexible Custom Type Plugin doesn't make sure the id is an integer, which allows users to perform XSS

Solution

Make sure the id of the post is an integer

Codes

"Cross-site scripting (XSS)", "Unsanitized data", "Sanitize data (enforce expected datatype)"

CVE-2011-5104

Context

Cross-site scripting (XSS) vulnerability in wpsc-admin/display-sales-logs.php in WP e-Commerce plugin 3.8.7.1 and possibly earlier for WordPress allows remote attackers to inject arbitrary web script or HTML via the custom_text parameter.

Problem

The custom-text parameter input is not escaped for html attributes which causes the potential for XSS

Solution

Escape the html attributes in custom-text parameter

Codes

"Cross-site scripting (XSS)", "Unsanitized data", "Not escaping characters"

CVE-2011-4926

Context

Cross-site scripting (XSS) vulnerability in adminimize/adminimize_page.php in the Adminimize plugin before 1.7.22 for WordPress allows remote attackers to inject arbitrary web script or HTML via the page parameter.

Problem

The user input in page parameter is not escaped which causes the potential for XSS

Solution

Escape user input in page parameter

Codes

"Cross-site scripting (XSS)", "Unsanitized data", "Not escaping characters"

CVE-2011-4803

Context

SQL injection vulnerability in wptouch/ajax.php in the WPTouch plugin for WordPress allows remote attackers to execute arbitrary SQL commands via the id parameter.

Problem

The id variable is not checked if it's an id number, which causes the potential to execute arbitrary SQL commands.

Solution

sanitize user input

Codes

"SQL injection", "Unrestricted access to database"

CVE-2011-4673

Context

SQL injection vulnerability in modules/sharedaddy.php in the Jetpack plugin for WordPress allows remote attackers to execute arbitrary SQL commands via the id parameter.

Problem

Based on exploit The id parameter is not checked if it's a valid id number, which causes the potential to execute arbitrary SQL commands

Solution

Saniitze User input

Codes

"SQL injection", "Unrestricted access to database"

CVE-2011-4671

Context

SQL injection vulnerability in adrotate/adrotate-out.php in the AdRotate plugin 3.6.6, and other versions before 3.6.8, for WordPress allows remote attackers to execute arbitrary SQL commands via the track parameter (aka redirect URL).

Problem

Based on exploit $wpdb->prepare can be misused to include a query, since no check is implemented on it.

Solution

Not found

Codes

"SQL injection", "Unrestricted access to database"

CVE-2011-4646

Context

SQL injection vulnerability in wp-postratings.php in the WP-PostRatings plugin 1.50, 1.61, and probably other versions before 1.62 for WordPress allows remote authenticated users with the Author role to execute arbitrary SQL commands via the id attribute of the ratings shortcode when creating a post. NOTE: some of these details are obtained from third party information.

Problem

In the rating attribute, the input provided by the user was not escaped which caused the potential for code injection

Solution

Escape the html attributes in the rating attribute

Codes

"Cross-site scripting (XSS)", "Not escaping characters", "Code Injection"

CVE-2011-4618

Context

Cross-site scripting (XSS) vulnerability in advancedtext.php in Advanced Text Widget plugin before 2.0.2 for WordPress allows remote attackers to inject arbitrary web script or HTML via the page parameter.

Problem

Characters received by the user for page parameter are not being escaped which causes the potential for XSS

Solution

Escape characters from user input

Codes

"Cross-site scripting (XSS)", "Unsanitized data", "HTML Injection", "Inject Javascript", "Not escaping characters", "Code Injection"

CVE-2011-4568

Context

Cross-site scripting (XSS) vulnerability in view/frontend-head.php in the Flowplayer plugin before 1.2.12 for WordPress allows remote attackers to inject arbitrary web script or HTML via the URI.

Problem

[VAGUE] the Requested URI from the Server was not being encoded, which created the option for users to XSS through this parameter.

Solution

Encode the requested URI

Codes

"Unsanitized parameter", "Cross-site scripting (XSS)", "Unsanitized data"

CVE-2011-4562

Context

Multiple cross-site scripting (XSS) vulnerabilities in (1) view/admin/log_item.php and (2) view/admin/log_item_details.php in the Redirection plugin 2.2.9 for WordPress allow remote attackers to inject arbitrary web script or HTML via the Referer HTTP header in a request to a post that does not exist.

Problem

In multiple files, characters provided by the user are not escaped which causes the potential for XSS

Solution

Escape characters from user input to disable XSS

Codes

"Cross-site scripting (XSS)", "Unsanitized data", "Escape user-supplied data", "Not escaping characters"

CVE-2011-4342

Context

PHP remote file inclusion vulnerability in wp_xml_export.php in the BackWPup plugin before 1.7.2 for WordPress allows remote attackers to execute arbitrary PHP code via a URL in the wpabs parameter.

Problem

A vulnerability has been discovered in the Wordpress plugin BackWPup 1.6.1 which can be exploited to execute local or remote code on the web server. The Input passed to the component "wp_xml_export.php" via the "wpabs" variable allows the inclusion and execution of local or remote PHP files as long as a "_nonce" value is known. The "_nonce" value relies on a static constant which is not defined in the script meaning that it defaults to the value "822728c8d9".

Solution

Not found

Codes

"Remote Code Execution", "Lack of sandbox", "inject code"

CVE-2011-3981

Context

PHP remote file inclusion vulnerability in actions.php in the Allwebmenus plugin 1.1.3 for WordPress allows remote attackers to execute arbitrary PHP code via a URL in the abspath parameter.

Problem

Attackers can execute PHP in header paths since there is no check if the file being attached exists locally and there is no escaping of characters.

Solution

Add a check to see if the file exists locally and escape certain characters.

Codes

"Remote Code Execution", "Lack of sandbox"

CVE-2011-1047

Context

Multiple SQL injection vulnerabilities in VastHTML Forum Server (aka ForumPress) plugin 1.6.1 and 1.6.5 for WordPress allow remote attackers to execute arbitrary SQL commands via the (1) search_max parameter in a search action to index.php, which is not properly handled by wpf.class.php, (2) id parameter in an editpost action to index.php, which is not properly handled by wpf-post.php, or (3) topic parameter to feed.php.

Problem

Based on exploit: https://www.exploit-db.com/exploits/16235/ The vulnerability exists due to failure in the "index.php" script to properly sanitize user-supplied input in "search_max" variable. Attacker can alter queries to the application SQL database, execute arbitrary queries to the database, compromise the application, access or modify sensitive data, or exploit various vulnerabilities in the underlying SQL database.

Solution

Sanitize user input

Codes

"SQL injection", "Unrestricted access to database"

CVE-2010-5295

Context

Cross-site scripting (XSS) vulnerability in wp-admin/plugins.php in WordPress before 3.0.2 might allow remote attackers to inject arbitrary web script or HTML via a plugin's author field, which is not properly handled during a Delete Plugin action.

Problem

The name and author field of a plugin could inject code which would be executed when trying to delete a plugin.

Solution

Escape correctly the strings of name and author fields when trying to delete a plugin, so that they're safe to be used in HTML.

Codes

"Cross-site scripting (XSS)", "HTML Injection", "Inject Javascript", "Not escaping characters", "Code Injection"

CVE-2010-4839

Context

SQL injection vulnerability in the Event Registration plugin 5.32 and earlier for WordPress allows remote attackers to execute arbitrary SQL commands via the event_id parameter in a register action.

Problem

Based on exploit The event_id parameter is not checked properly, which enables users to inject SQL commands through it.

Solution

Based on exploit: sanitize user input

Codes

"SQL injection", "Unrestricted access to database"

CVE-2010-4747

Context

Cross-site scripting (XSS) vulnerability in wordpress-processing-embed/data/popup.php in the Processing Embed plugin 0.5 for WordPress allows remote attackers to inject arbitrary web script or HTML via the pluginurl parameter.

Problem

Based on exploit: https://www.exploit-db.com/exploits/35066/ The Processing Embed plugin for Wordpress is prone to a cross-site-scripting vulnerability because it fails to properly sanitize user-supplied input in pluginurl parameter. An attacker may leverage this issue to execute arbitrary script code in the browser of an unsuspecting user in the context of the affected site. This can allow the attacker to steal cookie-based authentication credentials and launch other attacks.

Solution

Based on exploit: sanitize user input

Codes

"Cross-site scripting (XSS)", "HTML Injection", "Inject Javascript", "Code Injection"

CVE-2010-4518

Context

Cross-site scripting (XSS) vulnerability in wp-safe-search/wp-safe-search-jx.php in the Safe Search plugin 0.7 for WordPress allows remote attackers to inject arbitrary web script or HTML via the v1 parameter.

Problem

Based on exploit: https://www.exploit-db.com/exploits/35067/ The Safe Search plugin for Wordpress is prone to a cross-site-scripting vulnerability because it fails to properly sanitize user-supplied input in v1 parameter. An attacker may leverage this issue to execute arbitrary script code in the browser of an unsuspecting user in the context of the affected site. This can allow the attacker to steal cookie-based authentication credentials and launch other attacks.

Solution

Based on exploit: sanitze user input

Codes

"Cross-site scripting (XSS)", "Unsanitized data"

CVE-2010-3977

Context

Multiple cross-site scripting (XSS) vulnerabilities in wp-content/plugins/cforms/lib_ajax.php in cforms WordPress plugin 11.5 allow remote attackers to inject arbitrary web script or HTML via the (1) rs and (2) rsargs[] parameters.

Problem

Based on exploit: https://www.exploit-db.com/exploits/34946/ The cformsII plugin for WordPress is prone to multiple cross-site scripting vulnerabilities because it fails to properly sanitize user-supplied input. An attacker may leverage these issues to execute arbitrary

script code in the browser of an unsuspecting user in the context of the affected site. This may allow the attacker to steal cookie-based authentication credentials and launch other attacks.

Solution

Based on exploit: sanitize user data

Codes

"Cross-site scripting (XSS)", "Unsanitized data", "improper validation of user-supplied input"

CVE-2010-2924

Context

SQL injection vulnerability in myLDlinker.php in the myLinksDump Plugin 1.2 for WordPress allows remote attackers to execute arbitrary SQL commands via the url parameter.

Problem

Based on exploit The URL parameter is not validated properly, which enables users to inject SQL commands through it.

Solution

Not found

Codes

"SQL injection", "Unrestricted access to database"

CVE-2010-1186

Context

Cross-site scripting (XSS) vulnerability in xml/media-rss.php in the NextGEN Gallery plugin before 1.5.2 for WordPress allows remote attackers to inject arbitrary web script or HTML via the mode parameter.

Problem

Based on exploit. This vulnerability results from reflected unsanitized imput that can be crafted into an attack by a malicious user by manipulating the mode parameter of the xml/media-rss.php script. This vulerablity is triggered because the mode parameter on the media-rss.php script is not correctly escaped to avoid HTML code injection. $mode = $_GET["mode"]; This parameter is reflected back to the user if no correct mode is selected. } else { header('content-type:text/plain;charset=utf-8'); echo sprintf(__("Invalid MediaRSS command (%s).","nggallery"), $mode); exit; } It's worth to note that the Content-Type is chosen safely by the plugin, but this is not enough to avoid code injection because some browsers (most notably Microsoft Internet Explorer) choose the content type by parsing the content the web-server returns instead of obeying the proper headers.

Solution

Base on exploit: sanitize data

Codes

"Unsanitized parameter", "Cross-site scripting (XSS)", "Not escaping characters"

CVE-2010-0673

Context

SQL injection vulnerability in cplphoto.php in the Copperleaf Photolog plugin 0.16, and possibly earlier, for WordPress allows remote attackers to execute arbitrary SQL commands via the postid parameter.

Problem

The postid parameter is not validated properly, which causes the potential to inject SQL commands through it.

Solution

Not found

Codes

"SQL injection", "Unrestricted access to database"

CVE-2009-4748

Context

SQL injection vulnerability in mycategoryorder.php in the My Category Order plugin 2.8 and earlier for WordPress allows remote attackers to execute arbitrary SQL commands via the parentID parameter in an act_OrderCategories action to wp-admin/post-new.php.

Problem

Based on exploit My Category Order plugin does not check if the id for a query parameter is integer, which enables attackers to inject SQL commands

Solution

Based on exploit Check if the id is integer

Codes

"SQL injection", "Unrestricted access to database"

CVE-2009-4672

Context

Directory traversal vulnerability in main.php in the WP-Lytebox plugin 1.3 for WordPress allows remote attackers to include and execute arbitrary local files via a .. (dot dot) in the pg parameter.

Problem

Based on exploit Directory traversal can be performed through the pg parameter through '..' which may give users access to files.

Solution

Not found.

Codes

"Unsanitized data", "File path traversal"

CVE-2009-4424

Context

SQL injection vulnerability in results.php in the Pyrmont plugin 2 for WordPress allows remote attackers to execute arbitrary SQL commands via the id parameter.

Problem

Based on exploit The id parameter is not validated properly, which causes the potential for users to inject SQL commands through it.

Solution

Base on exploit: sanitize user input

Codes

"SQL injection", "Unrestricted access to database"

CVE-2009-2396

Context

PHP remote file inclusion vulnerability in template/album.php in DM Albums 1.9.2, as used standalone or as a WordPress plugin, allows remote attackers to execute arbitrary PHP code via a URL in the SECURITY_FILE parameter.

Problem

Based on exploit:

Security File parameter is not validated properly and can include php files, which causes the potential for arbitrary code execution

Solution

Not found

Codes

"Arbitrary code execution", "Unsanitized parameter"

CVE-2009-2383

Context

SQL injection vulnerability in BTE_RW_webajax.php in the Related Sites plugin 2.1 for WordPress allows remote attackers to execute arbitrary SQL commands via the guid parameter.

Problem

Based on exploit Guid parameter is not validated properly which enables users to perform SQL Injection

Solution

Not provided

Codes

"Sanitize data (enforce expected datatype)", "SQL injection", "Unrestricted access to database"

CVE-2009-2334

Context

wp-admin/admin.php in WordPress and WordPress MU before 2.8.1 does not require administrative authentication to access the configuration of a plugin, which allows remote attackers to specify a configuration file in the page parameter to obtain sensitive information or modify this file, as demonstrated by the (1) collapsing-archives/options.txt, (2) akismet/readme.txt, (3) related-ways-to-take-action/options.php, (4) wp-security-scan/securityscan.php, and (5) wp-ids/ids-admin.php files. NOTE: this can be leveraged for cross-site scripting (XSS) and denial of service.

Problem

Based on exploit: https://www.exploit-db.com/exploits/9110/

A vulnerability was found in the way that WordPress handles some URL requests. This results in unprivileged users viewing the content of plugins configuration pages, and also in some plugins modifying plugin options and injecting JavaScript code. Arbitrary native code may be run by a malicious attacker if the blog administrator runs injected JavasScript code that edits blog PHP code. No privileges are checked on WordPress plugins configuration PHP modules using parameter 'page' when we replace 'options-general.php' with 'admin.php'. The same thing happens when replacing other modules such as 'plugins.php' with 'admin.php'. Basic information disclosure is done this

way. For example, with the following URL a user with no privileges can see the configuration of plugin Collapsing Archives, if installed. http://[some_wordpress_blog]/wp-admin/admin.php?page=/collapsing-archives/options.txt

Solution

Not found.

Codes

"Inject Javascript", "Data leakage", "Lack of security checks"

CVE-2009-2122

Context

SQL injection vulnerability in viewimg.php in the Paolo Palmonari Photoracer plugin 1.0 for WordPress allows remote attackers to execute arbitrary SQL commands via the id parameter.

Problem

Based on exploit: https://www.exploit-db.com/exploits/8961/ Id parameter in plugin not validated, which enables users to inject SQL

Solution

Not provided

Codes

"Sanitize data (enforce expected datatype)", "SQL injection", "Unrestricted access to database"

CVE-2009-0968

Context

SQL injection vulnerability in fmoblog.php in the fMoblog plugin 2.1 for WordPress allows remote attackers to execute arbitrary SQL commands via the id parameter to index.php. NOTE: some of these details are obtained from third party information.

Problem

Based on exploit: https://www.exploit-db.com/exploits/8229/ Page id parameter is not validated properly, which enables users to inject SQL using it

Solution

Not provided

Based on exploit: sanitize data

Codes

"Unsanitized parameter", "SQL injection", "Unrestricted access to database"

CVE-2008-6811

Context

Unrestricted file upload vulnerability in image_processing.php in the e-Commerce Plugin 3.4 and earlier for Wordpress allows remote attackers to execute arbitrary code by uploading a file with an executable extension, then accessing it via a direct request to the file in wp-content/plugins/wp-shopping-cart/.

Problem

An e-Commerce plugin allows to upload executable files and to modify the we-shopping-cart directory which causes the potential for executing arbitrary code.

It is possible to upload a selected file to the ... /wp-content/plugins/wp-shopping-cart/ directory. If the directory is not writable (rare cases) you can use the insecure GET variable "imagedir" to directory traversal so you can upload in different directories.

Solution

Not provided.

Codes

"Remote Code Execution", "Unsanitized data", "File path traversal"

CVE-2008-5752

Context

Directory traversal vulnerability in getConfig.php in the Page Flip Image Gallery plugin 0.2.2 and earlier for WordPress, when magic_quotes_gpc is disabled, allows remote attackers to read arbitrary files via a .. (dot dot) in the book_id parameter. NOTE: some of these details are obtained from third party information.

Problem

Based on exploit book_id parameter can be used to traverse directories through ../ /wp-content/plugins/page-flip-image-gallery/books/getConfig.php?book_id=../../../../../../../../../../../etc/passwd%00123

Solution

Not found

Codes

"File path traversal"

CVE-2008-5695

Context

wp-admin/options.php in WordPress MU before 1.3.2, and WordPress 2.3.2 and earlier, does not properly validate requests to update an option, which allows remote authenticated users with manage_options and upload_files capabilities to execute arbitrary code by uploading a PHP script and adding this script's pathname to active_plugins.

Problem

WordPress is prone to a vulnerability that lets remote attackers execute arbitrary code because the application fails to sanitize user-supplied input. Attackers can exploit this issue to execute arbitrary PHP code within the context of the affected webserver process.

Solution

WordPress allows any user with manage_options capability to update directly any blog's option through wp-admin/options.php, so this feature can be used to perform (or hide) multiple attacks where WordPress expects safe data coming from the DB. This bug is very critical in those sites using WordPress MU, because any user has the manage_options capability.

Codes

"Arbitrary code execution"

CVE-2008-4625

Context

SQL injection vulnerability in stnl_iframe.php in the ShiftThis Newsletter (st_newsletter) plugin for WordPress allows remote attackers to execute arbitrary SQL commands via the newsletter parameter, a different vector than CVE-2008-0683.

Problem

Based on exploit The newsletter parameter is not validated properly, which causes the potential to execute arbitrary SQL commands. http://flymusic.co.uk/wp-content/plugins/st_newsletter/stnl_iframe.php?newsletter=-9999+UNION+SELECT+concat(user_login,0x3a,user_pass,0x3a,user_email)+FROM+wp_users--

Solution

Not found

Based on Exploit: sanitize input

Codes

"Sanitize data (enforce expected datatype)", "SQL injection"

CVE-2008-1982

Context

SQL injection vulnerability in ss_load.php in the Spreadsheet (wpSS) 0.6 and earlier plugin for WordPress allows remote attackers to execute arbitrary SQL commands via the ss_id parameter.

Problem

Based on exploit: https://www.exploit-db.com/exploits/5486/ The ss_id parameter is not checked if it's an integer and is included in query, which creates the option for SQL injection

Solution

Not found

Base on exploit: sanitize input

Codes

"Escape user-supplied data", "Sanitize data (enforce expected datatype)", "SQL injection"

CVE-2008-0491

Context

SQL injection vulnerability in fim_rss.php in the fGallery 2.4.1 plugin for WordPress allows remote attackers to execute arbitrary SQL commands via the album parameter.

Problem

Based on exploit: https://www.exploit-db.com/exploits/4993/ album parameter is not escaped or neutralized.

Solution

Not found

Based on exploit: escape parameter

Codes

"Escape user-supplied data", "SQL injection"

CVE-2007-5800

Context

Only WordPress installations on hosts which allow for register_globals = on allow_url_fopen = on in their php.ini settings are affected.

Multiple **PHP remote file inclusion vulnerabilities** in the BackUpWordPress 0.4.2b and earlier plugin for WordPress allow remote attackers to **execute arbitrary PHP code** via a URL in the bkpwp_plugin_path parameter to (1) plugins/BackUp/Archive.php; and (2) Predicate.php, (3) Writer.php, (4) Reader.php, and other unspecified scripts under plugins/BackUp/Archive/.

Problem

[BASED ON EXPLOIT] The URL parameter is **not validated properly and enables users to execute arbitrary PHP code**. This is an instance of CWE-98: Improper Control of Filename for Include/Require Statement in PHP Program ('PHP Remote File Inclusion')

Solution

Not found

Based on exploit: properly **validat**e url parameter

Codes

"Arbitrary code execution", "Unsanitized parameter"

CVE-2006-5705

Context

WP-DB-Backup allows the backup of the core WordPress database tables.

Multiple directory traversal vulnerabilities in plugins/wp-db-backup.php in WordPress before 2.0.5 allow remote authenticated users to **read or overwrite arbitrary files via directory traversal sequences** in the (1) backup and (2) fragment parameters in a GET request.

Problem

A **directory traversal** can be performed in backup and fragment parameters in GET requests, which enables users to **read or overwrite files through plugins**. [CWE-22: Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')]

Solution

Validate backup and fragment parameters to not allow directory traversal. In short, it verifies whether the GET parameters "backup" and "fragment" do not have traversal characters (such as ../ or ./ etc)

Codes

"File path traversal", "Lack of sandbox"