Risk ID	Technical Risk	Technical Risk Indicators	Related CVE, CWE or OSVDB IDs	Impact Rating	Impact	Mitigation	Validation Steps
1	Improper Neutralization of Directives in Dynamically Evaluated Code	moxie_dll.Moxiecode.Moxie silverlightmediaelement_dll.SilverlightMediaElement.MainPage	CWE ID 95	H	The software allows user-controlled input to be fed directly into a function (e.g. "eval") that dynamically evaluates and executes the input as code, usually in the same interpreted language that the product uses.	Validate all user-supplied input to ensure that it conforms to the expected format, using centralized data validation routines when possible. In general, avoid executing code derived from untrusted input.	Check that user input is validated properly to avoid dynamic code, discard if invalid code detected.
2	Improper Control of Filename for Include/Require Statement in PHP Program	www/wp-admin/plugins.php 151 www/.../wp-includes/template.php 501 www/.../wp-includes/template.php 503 www/wp-admin/update.php 90	CWE ID 98	H	The PHP application receives user-supplied input but does not properly restrict the input before using it in require(), include(), or similar functions. This can allow an attacker to specify a URL to a remote location from which the application will retrieve code and execute it.	Validate all user-supplied input to ensure that it conforms to the expected format, using centralized data validation routines when possible. Use white lists to specify known safe values rather than relying on black lists to detect malicious input.	Ensure that user supplied input is restricted to the right location access
3	SQL injection	www/board.php 30 www/includes/dblib.php 23 www/scoreboard/index.php 50 www/scoreboard/index.php 60 www/.../SimplePie/Cache/MySQL.php 344 www/wp-includes/wp-db.php 795 www/wp-includes/wp-db.php 797	CWE ID 89	H	This database query contains a SQL injection flaw. The function call constructs a dynamic SQL query using a variable derived from user-supplied input. An attacker could exploit this flaw to execute arbitrary SQL queries against the database.	Avoid dynamically constructing SQL queries. Instead, use parameterized prepared statements to prevent the database from interpreting the contents of bind variables as part of the query. Always validate user-supplied input to ensure that it conforms to the expected format, using centralized data validation routines when possible.	Make sure input sanitation is being performed to avoid dynamic construction of SQL attacks, discard if detected.
4	Use of Hard-Coded Password	www/board.php 15 www/board.php 18 www/includes/dblib.php 3 www/includes/dblib.php 6 www/scoreboard/index.php 31 www/scoreboard/index.php 34 www/scoreboard/index.php 111 www/scoreboard/index.php 114 www/.../network/site-new.php 74	CWE ID 259	M	A method uses a hard-coded password that may compromise system security in a way that cannot be easily remedied. The use of a hard-coded password significantly increases the possibility that the account being protected will be compromised. Moreover, the password cannot be changed without patching the software. If a hard-coded password is compromised in a commercial product, all deployed instances may be vulnerable to attack.	Store passwords out-of-band from the application code. Follow best practices for protecting credentials stored in locations such as configuration or properties files.	Check each application file for passwords and make sure they are removed. (Do not store plaintext passwords in the application)
5	Improper Neutralization of Script-Related HTML Tags in a Web Page	www/.../includes/ajax-actions.php 2748 www/board.php 43 www/board.php 44 www/board.php 58 www/board.php 59 www/board.php 64 www/.../class-ftp-pure.php 81 www/.../class-ftp-sockets.php 92 www/.../class-ftp-sockets.php 102 www/.../includes/class-ftp.php 228 www/.../class-phpmailer.php 631 www/.../wp-includes/class-smtp.php 168 .../class-wp-comments-list-table.php 536 www/.../class-wp-editor.php 1104 .../class-wp-list-table.php 564 .../class-wp-ms-users-list-table.php 189 www/.../js/customize-widgets.js 1042 .../customize-widgets.min.js 1 www/.../wp-admin/edit-tag-form.php 117 www/wp-admin/edit.php 311 www/.../wp-admin/includes/file.php 1097 www/.../wp-admin/js/image-edit.js 323 www/.../js/image-edit.min.js 1 www/wp-admin/import.php 116 www/scoreboard/index.php 119 www/wp-admin/install.php 176 www/.../wp-admin/load-scripts.php 161	CWE ID 80	M	This call contains a cross-site scripting (XSS) flaw. The application populates the HTTP response with user-supplied input, allowing an attacker to embed malicious content, such as Javascript code, which will be executed in the context of the victim's browser. XSS vulnerabilities are commonly exploited to steal or manipulate cookies, modify presentation of content, and compromise confidential information, with new attack vectors being discovered on a regular basis.	Use contextual escaping on all untrusted data before using it to construct any portion of an HTTP response. The escaping method should be chosen based on the specific use case of the untrusted data, otherwise it may not protect fully against the attack. For example, if the data is being written to the body of an HTML page, use HTML entity escaping; if the data is being written to an attribute, use attribute escaping; etc.	Check that each file performs user input sanitation to ensure no cross site scripting is present
6	Cleartext Storage of Sensitive Information in Memory	moxie_dll.Moxie.PngEncoder.DeflaterOutputStream	CWE ID 316	M	The application reads and/or stores sensitive information (such as passwords) unencrypted in memory, leaving it susceptible to compromise or erroneous exposure. An attacker with access to the system running the application may be able to obtain access to this sensitive data by examining core dumps and swap files, or by attaching to the running process with a debugger and searching mapped memory pages. Unless memory is explicitly overwritten, the sensitive information will persist until it is garbage collected and reallocated for other purposes.	Try to avoid storing sensitive data in plaintext. When possible, always clear sensitive data after use by explicitly zeroing out the memory. In languages that do not provide a mechanism for zeroing out memory, such as Java or C#, focus on minimizing the risk rather than eliminating it. Try to avoid using immutable types when handling sensitive information (for example, use a character array rather than a String). Keep the time window in which sensitive information is present in memory as short as possible to minimize the likelihood of it being swapped to disk.	Check that sensitive information is not stored in plain text. If it must be stored in plain text for some reason, make sure the memory used is zeroed or do not use immutable types.
7	Insufficient Entropy	moxie_dll.Moxiecode.MXI.Utils	CWE ID 331	M	Standard random number generators do not provide a sufficient amount of entropy when used for security purposes. Attackers can brute force the output of pseudorandom number generators such as rand().	If this random number is used where security is a concern, such as generating a session key or session identifier, use a trusted cryptographic random number generator instead. These can be found on the Windows platform in the CryptoAPI or in an open source library such as OpenSSL.	Check that a cryptographic random generator is being used
8	Missing Encryption of Sensitive Data	www/.../class-ftp-sockets.php 138 .../class-wp-filesystem-ftpext.php 68 .../class-wp-filesystem-ftpext.php 70	CWE ID 311	M	The application exposes potentially sensitive data by passing it into a function unencrypted. This could allow private data such as cryptographic keys or other sensitive information to be erroneously exposed.	Ensure that the application protects all sensitive data from unnecessary exposure.	Check that all sensitive data passed into functions is encrypted

Risk ID	Technical Risk	Technical Risk Indicators	Related CVE, CWE or OSVDB IDs	Impact Rating	Impact	Mitigation	Validation Steps
9	Use of a Broken or Risky Cryptographic Algorithm	www/.../wp-includes/bookmark.php 131 www/.../SimplePie/Caption.php 121 www/.../SimplePie/Category.php 103 www/.../includes/class-pclzip.php 2678 www/.../includes/class-pclzip.php 2716 www/.../class-phpass.php 67 www/.../class-phpass.php 70 www/.../class-phpass.php 139 www/.../class-phpass.php 141 www/.../class-phpass.php 144 www/.../class-phpass.php 146 www/.../class-phpmailer.php 1569 www/.../class-phpmailer.php 2044 www/.../class-phpmailer.php 2783 www/.../wp-includes/class-pop3.php 182 www/.../class-simplepie.php 720 www/.../wp-includes/class-smtp.php 416 www/.../wp-includes/class-smtp.php 424 www/.../class-snoopy.php 1215 www/.../class-wp-embed.php 192 .../class-wp-ms-themes-list-table.php 347 .../class-wp-plugins-list-table.php 473 www/.../class-wp-theme.php 203 www/.../class-wp-upgrader.php 1704 www/.../wp-includes/class-wp.php 379 www/wp-includes/comment.php 319 www/.../SimplePie/Copyright.php 93	CWE ID 327	M	The use of a broken or risky cryptographic algorithm is an unnecessary risk that may result in the disclosure sensitive information.	When there is a need to store or transmit sensitive data, use strong, up-to-date cryptographic algorithms to encrypt that data. Select a well-vetted algorithm that is currently considered to be strong by experts in the field, and use well-tested implementations. As with all cryptographic mechanisms, the source code should be available for analysis.	Check that all cryptographic algorithms used are not known to be easily broken
10	External Control of File Name or Path	www/.../class-wp-upgrader.php 1780 www/.../wp-admin/includes/file.php 65 www/.../wp-includes/ms-files.php 82 www/.../wp-admin/plugin-editor.php 63 www/.../wp-admin/plugin-editor.php 149 www/.../Text/Diff/Engine/shell.php 42 www/.../Text/Diff/Engine/shell.php 43 www/.../wp-admin/theme-editor.php 85 www/.../wp-admin/theme-editor.php 107 moxie_dll.Moxiecode.Com.Buffer	CWE ID 73	M	This call contains a path manipulation flaw. The argument to the function is a filename constructed using user-supplied input. If an attacker is allowed to specify all or part of the filename, it may be possible to gain unauthorized access to files on the server, including those outside the webroot, that would be normally be inaccessible to end users. The level of exposure depends on the effectiveness of input validation routines, if any.	Validate all user-supplied input to ensure that it conforms to the expected format, using centralized data validation routines when possible. When using black lists, be sure that the sanitizing routine performs a sufficient number of iterations to remove all instances of disallowed characters.	Check that each input sanitization checks against file path manipulation
11	Information Exposure Through an Error Message	www/board.php 18 www/includes/dblib.php 8 www/includes/dblib.php 27 www/scoreboard/index.php 34 www/scoreboard/index.php 114 www/wp-admin/plugins.php 284 www/.../network/themes.php 153	CWE ID 209	L	The software generates an error message that includes sensitive information about its environment, users, or associated data. The sensitive information may be valuable information on its own (such as a password), or it may be useful for launching other, more deadly attacks. If an attack fails, an attacker may use error information provided by the server to launch another more focused attack. For example, file locations disclosed by an exception stack trace may be leveraged by an attacker to exploit a path traversal issue elsewhere in the application.	Ensure that only generic error messages are returned to the end user that do not reveal any additional details.	Check error messages for sensitive information, and change those that provide unwanted access to that information
12	External Initialization of Trusted Variables or Data Stores	www/.../class-phpmailer.php 1050 www/.../class-phpmailer.php 1065 www/.../ID3/getid3.lib.php 602 www/.../ID3/getid3.lib.php 1356 www/.../wp-includes/ID3/getid3.php 190 www/.../wp-includes/ID3/getid3.php 1337 www/.../wp-includes/ID3/getid3.php 1349 www/.../Text/Diff/Engine/shell.php 50	CWE ID 454	L	A function is used to process options passed into a command line application. The optarg variable is used to store any additional arguments that an option requires. If optarg is used in an unbounded string copy, an attacker can specify overly long command line arguments and overflow the destination buffer, potentially resulting in execution of arbitrary code.	Be sure to limit the size of data copied from the optarg variable.	Check that all data copied from the optarg variable is limmited in size.