Risk ID	Technical Risk	Techical Risk Indicators	Related CVE, CWE, or OSVDB IDs	Impact Rating	Impact	Mitigation	Validation Steps
1	Improper Neutralization of Directives in Dynamically Evaluated Code ('Eval Injection')	plupload.silverlight.xap in System.Windows.Browser.ScriptObject get_EventTarget(), silverlightmediaelement.xap in void !ctor(System.Collections.Generic.IDictionary<string,string>)	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.	Do not allow untrusted input to be evaluated or otherwise interpreted as code. 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.	Attempt to send in code to the problem locations and verify that none of it is executed
2	Improper Control of Filename for Include/Require Statement in PHP Program ('PHP Remote File Inclusion')	www/wp-admin/admin.php	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.	Attempt to access routes with different permission levels and verify the correct pages are restricted/accessible
3	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	www/board.php, www/includes/dblib.php, www/scoreboard/index.php, www/.../SimplePie/Cache/MySQL.php, www/wp-includes/wp-db.php	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.	Attempt SQL injection by inserting an ' and verify that no error is thrown
4	Use of Hard-coded Password	www/board.php, www/includes/dblib.php, www/scoreboard/index.php, www/.../network/site-new.php	CWE ID 259	H	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.	Verify no passwords are stored plainly in application code through a quick search

5	Improper Neutralization of Script-Related HTML Tags in a Web Page (Basic XSS)	www/.../includes/ajax-actions.php, www/board.php, www/.../class-ftp-pure.php, www/.../class-ftp-sockets.php, www/.../includes/class-ftp.php, www/.../class-phpmailer.php, www/.../wp-includes/class-smtp.php,.../class-wp-comments-list-table.php, www/.../class-wp-editor.php, .../class-wp-links-list-table.php, .../class-wp-list-table.php, ../class-wp-ms-users-list-table.php, .../class-wp-posts-list-table.php, etc.	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. Both the OWASP ESAPI library for Java and the Microsoft AntiXSS library provide contextual escaping methods. For more details on contextual escaping, see https://www.owasp.org/index.php/XSS_%%28Cross_Site_Scripting%%29_Prevention_Cheat_Sheet. In addition, as a best practice, always validate user-supplied input to ensure that it conforms to the expected format, using centralized data validation routines when possible.	Attempt to enter <script> in problem areas and verify no code is executed
6	External Control of File Name or Path	www/.../Text/Diff/Engine/shell.php, void !ctor(System.IO.FileInfo)	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.	Attempt to access routes with different permission levels and verify the correct pages are restricted/accessible

7	Cleartext Storage of Sensitive Information in Memory	void set_Password(string)	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.	Verify no passwords are stored plainly in application code or tables through a quick search and design review
8	Information Exposure Through an Error Message	www/board.php, www/includes/dblib.php, www/scoreboard/index.php, www/wp-admin/plugins.php, www/.../network/themes.php	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.	Attempt to recreate errors and verify the error messages do not provide users with any more information than necessary
9	Insufficient Entropy	string guid(string)	CWE ID 331	L	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.	Verify the likelihood of collision from an attacker using a brute force method is low enough to ensure good security

10	Client Data Tampering	Cookie poisoning, localStorage poisoning	OSVDB ID 4934	L	May allow a remote attacker to steal the identity of an arbitrary user, including the administrator. The issue is due to the user cookie file storing the user name and encrypted password. If an attacker is able to steal a cookie file (via XSS or another method), they could edit their cookie file and use it to log in as an arbitrary user.	Verify cookies with authenication token for each session, make sure not to parse cookies without validation, and make sure not to set cookies with plaintext sensitive information	Review cookies using the developers console and verify sensitive information is not in plaintext. Test value changes in cookies to make sure application cannot be bypassed by altering client side
11	Use of a Broken or Risky Cryptographic Algorithm	www/.../SimplePie/Author.php, www/.../wp-includes/bookmark.php, www/.../SimplePie/Caption.php, www/.../class-phpass.php, www/.../class-phpmailer.php, etc.	CWE ID 327	L	The use of a broken or risky cryptographic algorithm is an unnecessary risk that may result in the disclosure of sensitive information.	Update cryptographic algorithms to community verified and accepted algorithms.	Verify that the algorithms set in place are those that are widely used in practice and were not self created.
12	FTP	nmap scan of site	OSVDB ID 88564 (closest match)	L	Insecure file transport protocol does not encrypt user data	Migrate to SFTP	Verify that there is no FTP service and is instead SFTP using an nmap scan

SOURCES: Veracode, OSVDB, NVD