risk ID	Technical Risk	Techical Risk Indicators	Impact Rating	Impact	Mitigation	Validation Steps
1	Code injection	Improper Control of Filename for Include/Require Statement in PHP Program ('PHP Remote File Inclusion') CWE ID 98 wp-admin/update.php @90	M (because it is fairly well contained by login)	Because the "$plugin" parameter is not sanitized or escaped, it could be abused to include malicious PHP code.	Sanitize the input, removing any patterns that could be used to include code from outside the plugins directory. This is already partially mitigated because it is functionality only available to users who have been granted access to install/update plugins	attempt to include a remote resource and verify that it is disallowed
2	SQL injection	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection') CWE ID 89 board.php @30 includes/dblib.php @23 scoreboard/index.php @50 scoreboard/index.php @60 wp-includes/SimplePie/Cache/MySQL.php @344 wp-includes/wp-db.php @795 wp-includes/wp-db.php @797	H	The impact of SQL injection has a lot to do with the permission of the databse user the application is using. If an application is using root credentials, the SQL injection can give the attacker access to the whole database instance (not just the database and tables for the application in question). Even a partially scoped DB user can create havoc, by issuing expensive queries, dumping all available information, and filling the tables with garbage data.	Validate/Sanitize all user input before including it as part of a SQL query. Specifically use something like mysql_escape_string(). Better would be to use a framework for making SQL calls that uses prepared statements Limit the access of your DB user accounts to only what then need. An application account shouldn't have the ability to drop tables unless it is an application requirement	attempt SQL injection with sqlmap or other tool
3	Hardcoded Passwords	Use of Hard-coded Password CWE ID 259 board.php @15 board.php @18 includes/dblib.php @3 includes/dblib.php @6 scoreboard/index.php @31 scoreboard/index.php @34 scoreboard/index.php @106 scoreboard/index.php @109	M	Hardcoded passwords are easily leaked in a variety of ways and are a pain to clean up after an account has been compromised, because you have to find instances and change them all.	Store credentials in config/settings files that are not stored in a web-published directory.	search for passwords in code.. verify that each time credentials are used, they are referenced from an external file
4	XSS	Improper Neutralization of Script-Related HTML Tags in a Web Page (Basic XSS) CWE ID 80 board.php @43 board.php @44 board.php @50 board.php @58 board.php @59 board.php @64 scoreboard/index.php @114 Occurences in WordPress code omitted	M	Application defacement is a serious threat with XSS. An attacker can inject javascript to be run in another users browser. Aside from defacement, XSS can break functionality, compromise data, and deny service	Sanitize all user inputs against <script> tags. This can be done in a variety of ways... You can just HTML escape or you can just drop requests that contain <script> tags	attempt to post <script> tags and verify that they're either dropped or neutralized
5	Missing Encryption	Missing Encryption of Sensitive Data CWE ID 311 Wordpress - FTP upload client	L	I believe this is only an issue if using WordPress's FTP upload mechanism... It has an SFTP option as well	Don't use FTP option (add warning about its vulnerabilty), or remove FTP option from code entiresly	Check that option is removed from UI
6	Broken/Risky Crypto	Use of a Broken or Risky Cryptographic Algorithm CWE ID 327 Listing of occurences omitted	L	md5 sum is used throughout Wordpress, but it is not used for hashing passwords. A case by case analysis of all 95 occurences would be needed to determine if any of these were of any real concern	replacing with a stronger hash/encryption if appropriate	check for remaining calls to md5sum or other weak hash/crypto
7	Path traversal	External Control of File Name or Path CWE ID 73	L	The cases in the WordPress code look like false positives. The variables used in constructing paths do not contain any user-supplied data	if it was really a problem, we'd sanitize this input and possibly use whitelists to restrict access to know locations	attempt directory traversal, if there actually was an input that was being passed to this funciton.

8	Information Leakage	Information Exposure Through an Error Message CWE ID 209 board.php @18 includes/dblib.php @8 includes/dblib.php @27 scoreboard/index.php @34 scoreboard/index.php @109	L	Error information passed to the client can show valuable information to an attacker. It can show instance specific information that will allow the attacker to focus his attack on a particular version or misconfiguration exposed through the error message.	swallow all error output to prevent it from making it to the user	create error conditions and verify that no error are displayed
9	Credentials passed in the clear	No SSL	H	Credentials passed in the clear are subject to sniffing	run services over https on port 443, redirect all port 80 traffic to 443	verify that all traffic is https
10	Denial of Service	nothing to prevent unlimited requests	L	If server is DOS'ed the service is offline. If the server is sufficiently DOSed, even legitimate admins will have a hard time connecting to try to fix the problem	use a caching system (like Varnish) to present cached copies of expensive pages use a rate limiter (like mod_evasive) to block any IPs or networks making too many requests	launch your own DOS and verify that it is blocked/dropped after reaching the intended threshold
11	Brute force login	None of the login pages do anything to prevent brute force login attemps.	M	excess server load leading to performance issues potential for account compromise	use application firewall (like ModSecurity) to identify and block excessive auth attempts from particular IPs or networks.	try to brute force with wpscan or some other tool and verify that the attack is blocked
12	weak passwords		M	cracked passwords can used to gain unauthorized access	enforce password complexity	verify that poor passwords are not allowed expire old passwords that are not known to comply with the complexity policy
13	shared credential	Same DB credentials are used for multiple applications	L	One compromised account gives an attacker broad access	use application specific credentials, with permissions/access rights narrowly scoped in accordance with the prinipcle of least privilege	verify each application has its own credentials
14	development leftovers	Source control information left on file system	L	The web publishing of .git and .svn folders can give an attacker an avenue to find the source code behind an application, which makes it much easier to mount a targeted attack	remove .git and .svn folders create .htaccess rules to prevent such files/directories from being served	attempt to load files from a .git folder