| Risk ID | Technical Risk | Technical Risk Indicators | Related CVE, CWE, or OSVDB IDs | Impact Rating |
|---|---|---|---|---|
| 1 | Hard coded username and password for database in file | board.php, line 15, 18; scoreboard/index.php lines 31, 34, 111, 114 | CWE-259 | H |
| 2 | SQL injection vulnerability | board.php, lines 30, 56, 62; scoreboard/index.php lines 50, 60 | CWE-89 | H |
| 3 | Improper control of filename for Include/Require statement in PHP | wp-admin/admin lines 238, 240; plugins.php line151; update.php line 90 | CWE-98 | H |
| 4 | Code injection by injecting untrusted input into an application that dynamically evaluates and executes the input as code (e.g. remote file includes and eval injection) | System.Windows.Browswer.ScriptObjectget_EventTarget() andSystem.Collections.Generic.Idictionary<string,string> | CWE-95 | H |
| 5 | Cross-site scripting; attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user. | Any of the game's web applications which used untrusted data in the output they generates without validating or encoding it. Found in over 100 lines of code, especially in board.php and anything with wordpress | CWE-80, CWE-79 | M |

| | | | | |
|---|---|---|---|---|
| 6 | Sensitive information is stored unencrypted in memory | Plupload silverlight; set-password(string) | CWE-316 | M |
| 7 | Insufficient entropy (random number generator too weak) | Plupload silverlight; string guit(string) | CWE-332 | M |
| 8 | Unencrypted sensitive data | class-ftp-sockets.php line 138, class-wp-filesystem-ftpext.php lines 68, 70 | CWE-311 | M |
| 9 | Using broken or risky cryptographic algorithms (such as MD5 or SHA-1) | 100 flaws, many in wordpress | CWE-327 | M |
| 10 | User is able to specify all or part of a filename, and thus they are able to fain unauthorized access to files on the server | class-wp-upgrader.php line 1780; shell.php lines 42, 43 | CWE-73 | M |
| 11 | Software generates an error message that includes sensitive information about its environment, users, or data | dblib.php lines 8, 27; scoreboard/index.php lines 34, 114 | CWE-209 | L |

|  |  |  |  |  |
|---|---|---|---|---|
| 12 | Critical internal variables or data stores initialized by the user | class-phpmailer.php lines 1050, 1065; getid3.lib.php lines 602, 1356; wordpress | CWE-454 | L |
| 13 | Extra services running | Netcat scan of the IP |  | M/H |

| Impact | Mitigation | Validation Steps |
|---|---|---|
| Attacker can get root privilege to the database | Do not hardcode passwords or usernames in an easily accessible file but rather store them out-of-band from the application code. Encrypt the password and use the hash to login, not the plaintext. | Check developer code for hard-coded passwords/usernames |
| Database information could be stolen or changed. Attacker could access the system as another user. | Sanitize and validate user-input (remove any special characters from use input). Avoid dynamically constructing SQL searches. | Automated static and dynamic analysis. Fuzzing, robustness testing, and fault injection. |
| Allows an attacker to execute code remotely including inserting files on the local machine and forcing that code to execute. | Validate all user-input and only allow certain, safe inputs. | Automated static analysis to test for input validation. |
| Arbitrary code can be executed. Injected code could access restricted data/files. Data could be stolen. | 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. | Automated static analysis to test for input neutralization. Check that programmers never use eval(). |
| Able to execute malicious code in the victim's browser. Attacker can steal cookies, modify content, and take confidential information. | Validate/sanitize user input. Restrict special characters, specifically html tags "<", ">" and "&". | Automated static analysis. Use the XSS Cheet Sheet (referenced by the CWE) which tests a variety of attacks specifically designed against weak XSS defense. |

| | | |
|---|---|---|
| Anything that is stored in the open, such as passwords, is susceptible to theft. | Do not store sensitive data without encryption. Minimize the time sensitive information is kept in memory, and clear that memory after use. | Automatic static and dynamic analysis. Manual analysis and checking. Using a language (like C) that has proper garbage collection/memory recycling that limits information staying in memory for longer than necessary. |
| Attacker could guess the random numbers generated and gain unauthorized access to a system if the random numbers are used for authentication or authorization. | Use a cryptographic random number generator (for things like session keys). | Code analysis for any standard random function used (using the built in rand library in PHP, Java, etc.). |
| Exposes potentially sensitive data (such as passwords) into a function unencrypted. If data is being transferred over a network, someone could sniff the network for this unencrypted data. | Protect all sensitive information via encryption and ensure information is not unnecessarily exposed to others. | Automated and manual analysis to measure the entropy of an input/output source (which may indicate the use or lack of encryption). |
| Any information encrypted with those algorithms is at risk of being exposed or stolen. | Use another algorithm, such as SHA-2 or SHA-3 | Automated or manual analysis looking for any use of broken functions. |
| Attacker could access or modify system files | Validate use input to ensure it conforms to the expected format. | Automated static analysis. Look for instances where any user input is used to interact with files. |
| Sensitive information may be exposed via error message. This information may be used for a later attack on the server. | Use only generic error messages | Manual and automated analysis. |

| | | |
|---|---|---|
| Attacker could gain access to and modify sensitive data or system information. Attacker could also use this for a buffer overflow attack, potentially resulting in execution of arbitrary code. | Don't allow users to initialize important variables and limit the size of data copied. | Automated static analysis. |
| Running an ftp service or having any sort of open port that could be easily broken into exposes possible user control of that computer or server; possibly even root access. | Do not leave unnecessary services running or open. Make sure to have strong passwords for any necessary services. | Run network scans, using something such as netcat, to make sure there are no unnecessary services. |