Jayme Woogerd
Technical Risk Analysis
November 14, 2014

ID	Technical Risk	Risk Indicators	Impact Rating	Impact	Mitigation	Validation Steps
1	SQL injection via improperly neutralized special elements	Evidence in static analysis of source code: board.php line 30; scoreboard/index.php line 50, 60	H	Application data can be read or modified; authentication for restricted data access can be bypassed	Use parameterized prepared statements for all constructed SQL queries; always validate user-supplied input	Prepared statements used for all dynamically constructed SQL queries
2	Improper control of filename in an ‘include’ or ‘require’ directive	Evidence in static analysis of source code: update.php line 90	H	An attacker could specify a URL to a remote location and the application would execute this (possibly malicious) code	Properly validate all user input to ensure that it conforms to the expected format; use a white list of known safe values for included files	Included filenames are validated against a known whitelist of safe files (validate by re-running static checker)
3	Use of hard-coded passwords	Evidence in static analysis of source code: board.php line 15, 16; scoreboard/index.php line 31, 34, 106, 109	M	Credentials much more likely to be compromised; password can't be changed without software patch;	Don't hardcode passwords, store them (properly) outside the application; follow best practices for credential storage	Code review or static checking yield no hardcoded passwords in source code
4	Cross-site scripting (XSS) via improperly neutralized HTML tags	Evidence in static analysis of source code: board.php line 43, 44, 50, 58, 59, 64; scoreboard/index.php line 114; many WordPress files	M	Attackers can embed malicious content, e.g. scripts to manipulate cookies, modify (deface) site content, and compromise confidential information	Validate (escape) all user input used to construct any portion of an HTTP response; customize escaping method to specific use case (e.g. HTML, attribute escaping)	All user input is properly sanitized (validate by re-running static checker)

Jayme Woogerd
Technical Risk Analysis
November 14, 2014

ID	Technical Risk	Risk Indicators	Impact Rating	Impact	Mitigation	Validation Steps
5	Missing encryption of sensitive data	Evidence in static analysis of source code: class-ftp-sockets.php line 138; class-wp-filesytem-ftpext.php line 68, 70	M	Private data such as cryptographic keys or other sensitive information could be exposed	Ensure all sensitive data is properly encrypted	All private/sensitive data is encrypted with current best practice methods (validate with static scan and code review)
6	Use of a broken or risky cryptographic algorithm	Evidence in static analysis of source code: 95 instances across many WordPress files	M	Private data such as cryptographic keys or other sensitive information could be exposed	Ensure all sensitive data is properly encrypted	All private/sensitive data is encrypted with current best practice methods (validate with static scan and code review)
7	External control of file name or path via directory traversal	Evidence in static analysis of source code: class-wp-upgrader.php line 1780 Text/Diff/Engine/ shell.php line 42 Text/Diff/Engine/ shell.php line 43	M	Attacker could gain unauthorized access to files on the server that are normally inaccessible to end-users	Validate user-supplied input; set file permissions appropriately; ensure server software is up to date and patched	Server software is patched to prevent directory traversal attacks; file permissions are set appropriately

Jayme Woogerd
Technical Risk Analysis
November 14, 2014

ID	Technical Risk	Risk Indicators	Impact Rating	Impact	Mitigation	Validation Steps
8	Information exposure through error messages	Evidence in static analysis of source code: board.php line 18; scoreboard/ index.php line 34; multiple Wordpress files	L	Error messages may expose sensitive data including the environment, users, or other data; attackers could use information like file locations from a stack trace to inform other attacks	Ensure only generic error messages are shown to the end-user that do not reveal any additional details	Rewrite error messages such that they are generic and do not reveal unnecessary details
9	WordPress admin password can be brute-forced	Increased server traffic; number of incorrect logins for accounts seen in logs	H	If successful attackers can gain full access to site to change or delete content; increased server traffic; degraded performance; possible denial of service	Lockout user account after 5 incorrect password tries	Account lockout flag set for user account on 5 incorrect password tries
10	External control of Git files via directory traversal		L	Attacker could gain unauthorized access to files on the server that are normally inaccessible to end-users; attackers could see source code or other sensitive data	Don't include the .git folder in a web page or application	Git folder is removed from application