Risk ID	Technical Risk	Technical Risk Indicators
1	Security by Obscurity	Inclusion of flag-related information in page, inclusion of unecessary information at the bottom of a page (e.g. logout.php and use of id parameter)
2	Cross Site Scripting	Client side input methods that don't get sanitized before use in web page. Board.php is rife with instances of this vulnerability
3	Open FTP Service	Netcat and nmap scan of IP reveals FTP among available services, while being non-intergral to webservice.
4	SQL Injection	Query to database made with un-sanitized user input (e.g. board.php, 20)
5	Credentials Management	Inclusion of password an user information hardcoed into source code. (e.g. www/board.php, 14 and 15, wp-config file and authetication keys)
6	Directory Traversal	Directory's traversed through the wp-uploads page
7	Use of Broken Cryptographic Hash	Use of MD5 hash function in creating new user credentials is not encouraged, due to known collisions and brute force assaults being effective in limited time.

8	Cookie Tampering	Main.php will load alternative pages if cookies are set, which can be tampered by users on the fly.
9	Weak Password	Hash for user 'bobo' can be cracked by using industry standard wordlists and tool 'wpscan' in 15 minutes or less.
10	Code Injection	Directory traversal allows for os command injections.

Related CVS CWE OSVDB IDs	Impact Rating	Impact
CWE-656: Reliance on Security Through Obscurity	M	Entirely dependent on the nature of the obscured information, which is part of the risk factor; obscured information can be anywhere to artifacts of old code to actual trade secrets. Big risk.
CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting') and CWE-80: Improper Neutralization of Script-Related HTML Tags in a Web Page (Basic XSS)	M	At least temporarily, websites can be rendered functionally useless if javascript is loaded automatically (as is common practice on most modern browsers). Fixing is simple though -- often a wipe of a database will do the job.
CWE-220: Sensitive Data Under FTP Root	M	Dependent on the nature of files in the ftp dump. Can be innocuous or can be debilitating.
CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	H	Anywhere from reveal of unpriviledged information to complete loss of information from the databases. Can be catastrophic.
CWE-798: Use of Hard-coded Credentials and CWE-259: Use of Hard-coded Password	M	If one person gains a glimpse at the right section of code, they're walking away with the keys to the kingdom.
CWE-78: Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	M	Allowing people the ability to traverse through directories they don't have direct access to can lead to larger issues in systems that rely (as this one does) on security by obscurity.
CWE-327: Use of a Broken or Risky Cryptographic Algorithm and CWE-759: Use of a One-Way Hash without a Salt	M	A sufficiently motivated cracker can break even the most secure of passwords within a 30 minute or so window with a robust GPU.

CWE-565: Reliance on Cookies without Validation and Integrity Checking and CWE-472: External Control of Assumed-Immutable Web Parameter	L	Cookie tampering is impactful insofar as the cookies are used to reveal sensitive information.
CWE-521: Weak Password Requirements	M	The impact of having a weak password should be self explanatory: editing privledges and information access beyond what any average user should have.
CWE-78: Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	H	Code injection into an system, depending on the sstem, can result in something as severe as a complete removal of all files from a host system, as well as infection through external malware.

Mitigation	Validation Steps
Remove information that isn't integral to application functionality, or provide steps that require user validation before revealing sensitive information instead of just obscuring it.	Difficult to validate. Promote as common practice the removal of old code or unnecessary information from code to be staged for deployment. As far as fixing goes, check your code over. And then check it over again. And again.
Input sanitization. Simple as that.	Barrage your application with a variety of XSS attacks. If they fail, you've done your job.
Shut down the service or require more austere authentication if the service is important.	Perform another scan to see if the service is still offered.
Input sanitization. Simple as that.	Toy around to try and disrupt information in the database, or to access data you don't have privleges for; if this cannot be done, you've done your job. Also use sqlmap as a tool to assist in your PenTest.
Remove hardcoding and include somewher separate from production-level code.	Ensure that no instances of your username and password remain in production level code. Also, enforce better coding practices moving forward with regard to the importance of password security.
Limit users from a directory traversal space to a series of pages displaying the intended content.	Attempt to move beyond the scope of reasonable directories, and hopefully fail to do so. Alternatively, confirm that you've removed any directory-traversing pages that exist.
Use a better hash (although this kind of falls on the fault of WordPress)	If the hash is removed and replaced with a more secure alternative (e.g. SHA-256 at the time of writing) then all should be well.

Make sure cookie's are checked for valid values, or remove reliance on them entirely.	Attempt to change cookies using a request-manipulation tool, and see if similar changes from earlier produce pages that still contain sensitive information
Set up more stringient and impactful password restrictions on new users and force old users to update.	Attempt to bruteforce crack a password/hash of an account, and hopefully don't succeed within the first few hours.
Sanitize user input and remove instances of 'eval'.	Confirm that all instances of 'eval' are removed from production level code and that more care is taken in performing system operations.