Technical Risk Analysis

Risk ID	1
Technical Risk	User-controlled input is fed directly into a function – this can lead to remote code injection.
Technical Risk Indicators	Source for client-side javascript has changed, and this change is not reflected in any staff code-update logs.
Impact Rating	Very High
Impact	Redirection of users to hostile web-sites, stealing of cookie or session information, installing malware via users' browsers, attacking browser extensions, and more
Mitigation	Avoid executing user input as code. Validate all user-supplied input to ensure it conforms to the expected format. Sanitize your input.
Validation Steps	Reject user input if <scrpit> flag is included. Implement centralized data validation routines.

Risk ID	2
Technical Risk	Remote Code injection – user input is not sanitized before used in require(), include() or similar functions. An attacker could cause the application to retrieve and execute code from a remote URL.
Technical Risk Indicators	Traffic has been routed from your website to a hostile one, malignant code being executed on user's browsers
Impact Rating	High
Impact	Redirection of users to hostile web-sites, stealing of cookie or session information, installing malware via users' browsers, attacking browser extensions, and more
Mitigation	Sanitize input, ensure that user input conforms to expected style. Use white lists to specify known safe values, rather than relying on black lists to pick out unsafe ones. Attackers can be creative. Alternatively, implement other PHP functions that are less vulnerable.
Validation Steps	Words that are not found on the white list result in input being rejected. Other functions pass a static code analysis.

Risk ID	3
Technical Risk	SQL Injection – a SQL query is made using user-supplied input. This URL input query could be written to extract unauthorized information from the table.
Technical Risk Indicators	The classified or private information stored in the database is being utilized by unauthorized users. For instance, you notice the same machine loging in as an exorbitant number of users.

Technical Risk Analysis

Impact Rating	High
Impact	An attacker can execute administrative operations on the database to retrieve, delete, or alter stored information.
Mitigation	Use prepared statements rather than dynamically constructing SQL queries. i.e. give users a choice of options rather than open access. Cross-reference user input with white list of accepted input strings. Normalize all user-supplied data before submitting the query; ex: URL or HTML encoded so that characters such as ' become altered or appended to.
Validation Steps	User input with specific characters is rejected, or at least does not result in access to sensitive data.

Risk ID	4
Technical Risk	Password stored in application code – can be viewed by attacker by looking at source code
Technical Risk Indicators	Authorizations with passwords in the code are being accessed more frequently than ideal.
Impact Rating	Medium
Impact	Attackers can read source code, view the password after the word “secret” or “password”, and access insecure information using the password.
Mitigation	Store passwords out-of-band from application code; for instance, in configuration or properties files. Otherwise, encrypt these passwords in the code.
Validation Steps	Grepping source code for passwords or keywords yields no results

Risk ID	5
Technical Risk	Cross Site Scripting – Script-related HTML Tags in Web page are not properly neutralized.
Technical Risk Indicators	Users report site leaking information or suspicious activity occurring. Multiple hits and connections to suspicious web sites noticed.
Impact Rating	Medium
Impact	Redirection of users to hostile web-sites, stealing of cookie or session information, installing malware via users' browsers, attacking browser extensions, and more
Mitigation	Screen user supplied input through white lists. Use output filtering to sanitize all output generated by user-supplied input. Sanitize inputs, look for HTML tags and reject user input if found.
Validation Steps	Input with HTML tags rejected, scripts from outside sources rejected.

Technical Risk Analysis

Risk ID	6
Technical Risk	Cryptographic Issues, sensitive information stored in plain text in memory, or encrypted using broken or overly-simplistic algorithms.
Technical Risk Indicators	Information that can be retrieved by the passwords found in memory dumps or source code being accessed. Memory dups and swap files being examined.
Impact Rating	Medium
Impact	Passwords stored in system memory can be accessed and exploited.
Mitigation	Do not store sensitive data in plaintext in memory. Either encrypt it in memory, or store it outside of the source code or un-safe memory.
Validation Steps	Grepping for passwords in the source code yields no results

Risk ID	7
Technical Risk	Sensitive data passed unencryped to a function, may be exposed
Technical Risk Indicators	Data from these functions is being accessed and exploited
Impact Rating	Medium
Impact	Passwords passed as function arguments can be accessed and exploited
Mitigation	Ensure that application protects all sensitive data from unnecessary exposure
Validation Steps	Can't find passwords passed in unsafe sunctions

Risk ID	8
Technical Risk	Attackers can brute force the ourpur of pseudorandum number generators to match randomly generated access indicators, such as session ids.
Technical Risk Indicators	Many log in attempts seen in server log, possible DDOS attack symptoms.
Impact Rating	Low
Impact	Session tokens can sometimes result in access, exposing user or system access & information
Mitigation	If a random number is necessary, use a safe cryptographic number generator. Reject specific user accounts if too many attempts noticed
Validation Steps	User accounts locked after multiple attempts to match session ID.

Risk ID	9

Technical Risk Analysis

Technical Risk	Insufficient or broken cryptographic algorithm used
Technical Risk Indicators	In the event that an attacker is trying to brute force test passwords poorly encrypted, you will notice a significant number of failed attempts to log in.
Impact Rating	Medium
Impact	Passwords or sensitive information can be cracked using known decryption algorithms, or possibly by simple brute force.
Mitigation	Switch to trusted Encryption algorithms
Validation Steps	Attempting to decrypt passwords takes an exorbitant amount of time

Risk ID	10
Technical Risk	Directory traversal – paths to unintended folders and files can be input by user
Technical Risk Indicators	Server log sends multiple 404 error requests for extensions of URL ex: mysite.com/passwords. Hopefully these are not 200 requests for unintended directories
Impact Rating	Medium
Impact	Directories can be accessed by attackers and sensitive information exposed. This can even include source code, which could perhaps be edited
Mitigation	Sanitize all user input, espescially on the navigation bar. Limit which folders can be accessed without proper permission. If possible, remove unnecessary pages from the site. Block users if too many false directories are attempted
Validation Steps	Attempts to access the specific directory from unauthorized users fails. User is blocked after a significant number of directory access attempts.

Risk ID	11
Technical Risk	Information Leakage in Error Message
Technical Risk Indicators	Examining the server log shows that errors are frequently caused by specific users
Impact Rating	Low
Impact	Passwords or other sensitive information in Error message can be exploited or accessed.
Mitigation	Make error messages as generic as possible, and ensure that no sensitive information is displayed.
Validation Steps	Generation of error messages shows no unnecessary information

Risk ID	12

Technical Risk Analysis

Technical Risk	Initialization occurs using external variables
Technical Risk Indicators	Application is opened or initialized repeatedly, with the external input changed only slightly or resembling a word list
Impact Rating	Medium
Impact	Application may assume that the input cannot be tampered with, and misuse it in a harmful way
Mitigation	Initialize the program using pre-selected input. Specifically, limit the size of data copied from the optarg variable.
Validation Steps	External variables have no effect on how the page is initialized

Risk ID	13
Technical Risk	Cookie manipulation leads to improper access – session ID is stored in a cookie that can easily be changed to true. (more wrong)
Technical Risk Indicators	Server log displays unexpected users accessing sensitive directories or information
Impact Rating	Medium
Impact	Attackers can gain access to administration pages, and from there manipulate the site itself or access information
Mitigation	Store session ids on the server, rather than client side. Additionally, use trusted algorithms to generate these ids.
Validation Steps	Manipulation of locally stored data does not lead to unauthorized access.

Risk ID	14
Technical Risk	Sensitive data stored in local variables (ngtnw)
Technical Risk Indicators	Passwords stored in local variables are being exploited
Impact Rating	High
Impact	Attackers can gain access to administration pages, and from there manipulate the site itself or access information
Mitigation	Store this sensitive information server side, only store uncritical information on client-side.
Validation Steps	Examination of local variables leads to no passwords being found