Technical Risk Analysis Table for CTF

ID	Technical Risk	Indicators	Impact Rating	Impact	Mitigation	Validation Steps
1	Eval Injection	Function eval() appears in the codes (i.e. in the file plupload.silverlight.xap). Malicious codes were run on the server.	High	Allows user-controlled input to be fed directly into eval()and malicious codes can be executed on the server.	Eliminate the usage of eval(). If cannot, perform user input validation and filtering to prevent code injected to eval().	Try to input malicious codes to see whether it is executed. If it happens, try to mitigate it.
2	Code Injection (Remote File Includes)	The PHP application receives user-supplied input but does not properly restrict the input before using it in require().(i.e. in the file www/wp-admin/update.php). Malicious codes were run on the server.	High	Allows an attacker to specify a URL to a remote location from which the application will retrieve code and execute it.	Validate all user-supplied input to ensure that it conforms to the expected format. Use white lists to specify known safe URL.	Try to input URL with malicious codes to see whether the codes will be retrieved and executed. If it happens, try to mitigate it.

3	SQL Injection	Logs show a lot of improper login activity, and SQL query with strings like " or '1'='1". Database tables are altered or dropped by attackers.	High	Allows attackers to manipulate database queries in order to access, modify, or delete arbitrary data.	Use parameterized prepared statements rather than dynamically constructing SQL queries. Validate user-supplied input using positive filters to ensure that it conforms to the expected format.	Try to input SQL injection codes into the login form and get improper access to the database, performing data altering or deletion and see whether it works. If so, try to mitigate it.
4	Sensitive Information Direct Access	File storing sensitive information can be directly access by URL and without any authentication. (i.e. flag.txt).	High	Allows attackers to access sensitive information that not expected to be disclosed to unauthenticated users.	Eliminate direct mapping of URL to files storing sensitive data, as well as use access control for these resources or files.	Try to use URLs to directly access the files with sensitive data. Can even perform directory traversal on them to see whether the files can be accessed. If so, try to mitigate it.
5	Packed-refs Direct Access	Git packed-refs can be directly access by URL: http://67.23.79.113/git/packed-refs	Medium	Allows attackers to access information on your git repository that not expected to be disclosed to unauthenticated	Do not put git folder on the website.	Try to use URLs to directly access the git files. Eliminate it if the access is allowed.

				users.		
6	Directory Traversal	Path manipulation flaw in source codes , in which the argument to the function is a filename constructed using user-supplied input. Improper user input including path traversal substrings are found in logs.		May enable an attacker to access or modify otherwise protected system resources that would normally be inaccessible to end users.	Validate all user-supplied input and filter all disallowed contents to ensure that it conforms to the expected format.	Review the source codes and make sure the validation and filtering will be done in each necessary places.
7	Cookie Tampering	Users modified cookies to attain access to some sensitive information.	High	Allows attackers to modify the value of a cookie before sending them back to the server, enables attackers to carry out some sort of attack against the server that relates to some sort of data contained in the cookie.	Only store a randomly generated unique session identifier in the cookie, and everything else (especially the business-crucial and sensitive data) is stored on the server.	Review the source codes and make sure no codes will result business-crucial and sensitive data storing in client cookies.
8	Brute-force Password Breaking	User authentication to system can be broken by brute force. Number of	High	Allows attackers get the privileges of the authenticated users. Number of incorrect	Lock out user account on 5 incorrect password tries by setting	Account lockout flag set for user account on 5 incorrect

		incorrect logins for accounts seen in logs; performance of login server has been degrading.		logins for accounts seen in logs; performance of login server has been degrading. Possible denial of service.	account lockout flag to true.	password tries.
9	File extension hidden	run.exe file downloaded from the website is actually pcap file.	High	May cause unexpected file type with malicious codes opened and executed.	One method is to use “file” command to know what the file really is before open it.	Make sure to check the file type before opening it.
10	Use of Hard-coded Password	Hard-coded passwords appear in source codes. (i.e. in the file www/board.php)	Medium	Significantly increases the possibility that the account being protected will be compromised. Moreover, the password cannot be changed without patching the software. If the hard-coded password is compromised, all deployed instances may be vulnerable to	Store passwords out-of-band from the application code. Follow best practices for protecting credentials stored in locations such as configuration or properties files.	Review the source codes and make sure there is no password hard coded in the source codes.

				attack.		
11	Cross-Site Scripting	Improper Neutralization of Scipt-Related HTML Tags in a Web Page found in 75 places in source codes (populate the HTTP response with user-supplied input). Attacks exploiting this vulnerability are successfully applied on the application. (i.e. Javascript alerts)	Medium	Allows attackers to embed malicious content (i.e. Javascript codes) steal or manipulate cookies, modify presentation of content, and compromise confidential information.	Validate user-supplied input using positive filters (white lists); Use output filtering to sanitize all output generated from user-supplied input; Do not permit users to include HTML content in posts, notes, or other data that will be displayed by the application. If users are permitted to include HTML tags, then carefully limit access to specific elements or attributes, and use strict validation filters to prevent abuse.	Try to do HTML or Javascript code injection to the website. If it succeeds, try to mitigate it.
12	Cleartext Strorage of	The application reads and/or stores	medium	An attacker with access to the system	Try to avoid storing sensitive data in	Access to the system running

	Sensitive Information in Memory	passwords unencrypted in memory (when set_Password(string) is used in Module plupload.silverlight.xap). Passwords are exposed to attacker who access to the system.		running the application may be able to obtain access to this sensitive data by examining core dumps and swap files, or by attaching to the running process with a debugger and searching mapped memory pages.	plaintext. When possible, always clear sensitive data after use by explicitly zeroing out the memory. Keep the time window in which sensitive information is present in memory as short as possible.	the application, examine core dumps and swap files, or by attaching to the running process with a debugger and searching mapped memory pages.
13	Missing Encryption of Sensitive Data	Exposes potentially sensitive data by passing it into a function unencrypted.	Medium	Could allow private data such as cryptographic keys or other sensitive information to be erroneously exposed.	Ensure that the application protects all sensitive data from unnecessary exposure.	Review the source codes and make sure this flaw is eliminated.
14	Use of a Broken or Risky Cryptographic Algorithm	The usage of a broken or risky cryptographic algorithm for the sensitive information is found in 95 places in the source codes.	Medium	May result in the disclosure of sensitive information.	Use a more sophisticated cryptography that has not been broken for encrypting sensitive data.	Review the source codes and make sure no usage of a broken or risky cryptographic algorithm for the sensitive

						information.
15	Information Exposure Through an Error Message	When user authentication failed, the application generated an error message that includes sensitive information about its environment and associated data.	Low	The sensitive information may be valuable information on its own (such as a password), or it may be useful for launching other more deadly attacks.	Ensure that only generic error messages are returned to the end user that do not reveal any additional details.	Try to get to the error page and see whether there is sensitive information exposed on the page.