Michael Tran Comp116 A4

Risk ID	Technical Risk	Technical Risk Indicator	Impact Rating	Impact	Mitigation	Validation Steps
1	Path Traversal	-http://67.23.79.113/flag.txt is not reachable through website navigation, but is still accessible.	H	Attacker may be able to access critical files. Confidential files can be accessed. Critical files such as programs, libraries, or other important data can be accessed.	Neutralize special elements within the path name that can resolve a location in a restricted directory. Do not make such directories public.	Confirm that the page is no longer accessible through HTTP GET.
2	Improper Neutralization of Directives in Dynamically Evaluated Code ("Eval Injection")	silverlightmediaelement.xap plupload.silverlight.xap	H	Silverlight applications are susceptible to code injection attacks. slinject can inject assembly into any Silverlight process. The software can receive input from an attacker that is not neutralized before being evaluated. (e.g. "eval")	Remove silverlight, assume all input is bad and use a whitelist to allow input to this particular application. Remove eval().	Try to refactor so that eval or in this case, silverlight is no longer necessary. If it can be done you should be safe.
3	Improper Neutralization of Script Related HTML tags in a web page (Basic XSS)	users.php themes.php load-scripts.php	H	Multiple instances where script elements are not neutralized in html tags. This leaves the pages vulnerable to attackers who can enter data that is then sent to other users without being validated.	Check user input against a white list, a filter. Create a filter against well-known XSS attacks. Sanitize input being on the page such as the URL, GET and POST parameters, cookie data, header data, and referrer objects.	Pen test the page using a large variety of XSS vectors. Though no filter is guaranteed safe, confirm that the page is safe against common exploits.
4	Cleartext Storage for Sensitive Information / Missing Encryption for Sensitive Data	- http://67.23.79.113/.git/packed-refs has a plaintext key on the page (also a path traversal exploit - http://67.23.79.113/.git/logout.php	L	Attacker can see sensitive data easily by looking at source code.	Either encrypt sensitive data or remove it from public pages if it is in fact sensitive.	Easily fixed by either removing the data or encrypting it. This is verified by visiting the page and confirming that the data is not present anymore.
5	Use of Broken or Risky Cryptographic Algorithm	-"encrypting" the key by making it a pdf called "fun.exe" is easily decrypted -http://67.23.79.113/ngtnw	M	Confidentiality of data is compromised. Attacker can modify sensitive data using the same cryptographic algorithm.	Manually pen test to find any weak algorithms used. Change the crypto algorithm to a more secure methods	Confirm that the new algorithm used is not broken.
6	Improper Neutralization of Special Elements in Sql Command ("SQL Injection")	-SQL injection on the public board. e.g. "or 1=1" can be used to reveal more information on the board -The /admin.php page is vulnerable to SQL injection which can be used to login and access secure information.	H	The handling of external SQL input commands does not neutralize special elements. Confidential data is no longer confidential. SQL based access control is broken and user authentication is no longer valid. Data integrity is not guaranteed as attackers can directly change or delete information in the SQL database.	-Use prepared statements that allow the database to distinguish between code and data. These statements ensure that the attacker cannot adjust the query to be a command. -Escape all user supplied input, this is more valid in many cases such as login as special characters should not be allowed and prepared statements would not be effective given a large amount of possible user input. e.g. username and passwords	Escape the user input and do not allow special characters such as single ticks in either the user name or the password. This will reject any user attempts at SQL injection whenever in login forms.
7	Brute Force Password Cracking	http://67.3.79.113/wp-login.php	M	Broken Authentication and	-Use stronger passwords that are	Confirm that new passwords

		can be brute forced. For example { wpscan --url * --wordlist } cracks the password within seconds.		Session Management. Attackers can compromise passwords and other flaws to assume other user identities.	not as easily bruteforced. Force any users to comply to password standards which ensure the password security.	satisfy requirements and cannot be bruteforced using wordlists in any reasonable amount of time.
8	Cookie Poisoning	- 67.23.79.113/admin.php after login with SQL injection cookies can be set to fake successful login	H	-Broken Authentication System, does not verify credentials or session correctly	-Use one session token which references server side data. - Use hidden forms - Verify the cookie value against the database to check if it is valid for the user.	Confirm that important cookies that handle user authentication are visible and editable by end users.
9						