Michael Silverblatt, Prof. Ming Chow, Comp 116 - Computer System Security
Technical Risk Analysis Table - CTF Game

	Risk ID	Technical Risk	Technical Risk Indicators	Impact Rating	Impact	Mitigation	Validation steps
SQL Injection	1	Users can force malicious SQL requests to the database because of insufficient input validation.	During CTF game, flags and other information were found via dumping of posts and users tables via SQL Injection.	H	Vulnerability of potentially sensitive information. Credentials and posts can be viewed by unauthorized users.	Use regular expressions to allow only certain ascii values in input, filter out any “code-like” symbols. Log SQL database access, periodically scan for unauthorized attempts.	Attempts to inject SQL through forms or the URI result in error pages or correctly while ignoring malicious code.
Cross-site Scripting (XSS)	2	Improper neutralization of scripts in user-supplied post content.	Adding code within script tags into posts and submitting them caused the code to be executed on page load.	M	Ranges from annoying pop-ups and ads to serious security flaws when important information is being handled on the front end.	Validate all input for common code symbols like “<“ or “>” (not only input fields which are expected).	Adding javascripts to posts and other parts of the application lead to error messages or displaying of content in ascii text through correct character transformations.
Code Injection	3	User-generated code can be executed on the server through unfiltered ‘eval()’ statements.	Adding php code (such as phpinfo();) to the id parameter in the URI caused it to be executed.	H	A user can gain access to a multitude of unauthorized information if they can execute code on a server, not limited to the /etc/password file and other vital system information.	Ideally, refactor code such that eval statements are unnecessary. If that is impossible, use a whitelist of acceptable input for any user-submitted field that will be included in an eval statement.	Anything other than an integer value for the id parameter is treated as invalid input.

	Risk ID	Technical Risk	Technical Risk Indicators	Impact Rating	Impact	Mitigation	Validation steps
Information Leakage	4	Information about the system is leaked to the user through error messages.	When a connection to the database fails, a generic mysql error message is displayed.	M	Knowing more about a system will allow malicious users to use targeted and known exploits to disrupt that system. A MySQL error messages makes an application far more vulnerable to SQL Injection.	Remove any specific information from error messages.	On database error, simple error pages such as a 500 Internal Server Error page.
Credentials Management	5	Credentials stored on server in plain text hard-coded.	If source code is available, it is simple to find comparisons to fixed strings. Otherwise, use manual penetration testing or use monitoring tools to extract code from a running program and look for fixed string comparison	H	If password is leaked, anyone has access to server. If the software is distributed, changing passwords could be very difficult or impossible and allow for worms. If front end systems use the hard coded passwords, they can easily be obtained.	Store login credentials in a remote keychain. Use encryption wherever possible to add a layer of security.	Monitoring tools reveal nothing that resembles fixed-string comparison, manual review of source code finds nothing.
Steganography	6	Information can be hidden in images within the application.	A version of the logo image that is downloaded on page load contains potentially sensitive information stored within it.	L	Potentially important information or code can both be stored by users without the server's knowledge, or secret information on the server can be viewed by unauthorized users.	Encrypt any sensitive information, scan any filetypes known to be vulnerable to steganography for data in empty spots, checksum all data when possible.	Hidden information cannot be accessed without encryption key, hidden information cannot be added to files uploaded to server.

	Risk ID	Technical Risk	Technical Risk Indicators	Impact Rating	Impact	Mitigation	Validation steps
Cookie Tampering	7	Credential system can be bypassed by changing a value in a locally stored cookie.	Changing the cookie value while providing a password known to be incorrect allows access to the system.	H	Anyone can gain access to the system.	Perform validations on all cookies on server side, use temporary session keys rather than a simple yes/no in a cookie.	Changing cookie values does not allow access to application, correct password is required.
Weak Passwords	8	Server username:password combinations can be cracked using common online hash decryptors.	Password hash for account pheller can be looked up in free online rainbow tables.	H	Anyone with access to password hashes (e.g. through code injection) can decrypt them and access the server.	Implement a stricter security policy for all users/administrators requiring that all passwords meet certain length and character criteria, as well as not being on a list of x most common passwords.	Password hashes cannot be decrypted via rainbow table lookup or brute force within a reasonable amount of time.
Buffer Overflow	9	Stack buffer can be overwritten, leading to crashes, infinite loops or running of malicious code.	Namegame found in runme.c can be overflowed because it uses strcpy instead of strncpy.	M	Anyone who gains access to an account with executable access to the namegame will be able to execute arbitrary code by overflowing the buffer.	Always use “n” string functions in C programs to avoid buffer issues.	Namegame refuses strings above a certain lengtth, only copies first n bytes.