RISK ID	TECHNICAL RISK	TECHNICAL RISK INDICATORS	RELATED CVE, CWE, or OSVDB IDs	IMPACT RATING	IMPACT	MITIGATION	VALIDATION STEPS
1(Code Injection)	Code injection is the process of injecting untrusted input into an application that dynamically evalutes and executes the input as code.	if the input doesn't conform to the expected format	CWE-95, CWE-98	VH/H	Host Takeover, website defacement, loss of integrity/accountability	secure and sanitize and validate inputs	after a few tries of invalid input, lock the user out
2(SQL Injection)	SQL injection vulnerabilities occur when data enters an application from an untrusted source and is used to dynamically construct a SQL query	if the input looks like malicious SQL code	(CWE ID 89)	H	Attacker gets access to database and can manipulate it	validate and sanitize all user inputted data, have good error messages that don't give away too much information	You can't get database information when injecting SQL statements

3(Credentials Management)	Improper management of credentials, such as usernames and passwords, may compromise system security. In particular, storing passwords in plaintext or hard-coding passwords directly into application code.	discovering plain text passwords in the source code	CWE ID 259	M	attacker can access credentials of other users and pretend to be anyone	use a cryptographic one-way hash for data stored outside of the application code. Don't store passwords in easily accessible locations	two-step authentication, additional security question in order to login
4(Cross Site Scripting)	when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user	in the input looks like malicious javascript / script code	CWE ID 80	M	manipulate cookies, deface websites, and compromise sensitive information	sanitize inputs and validate that the input doesn't have <> and/or <script> tags	website content is unmodified when a user tries to input special characters such as <script> tags

5(Cryptographic Issues)	Common cryptographic mistakes include, selecting weak keys or weak cipher modes, unintentionally exposing sensitive cryptographic data, using predictable entropy sources, and mismanaging or hard-coding keys.	key information is exposed, and/or poorly chosen cryptography methods	CWE ID 316, CWE ID 331, CWE ID 311, CWE ID 327	M	attacker can gain access to sensitive information that was meant to be secret	don't write your own crypto! Use well known crypto functions, don't expose private key information	sensitive data is no long being passed around before being encrypted first
6(Directory Traversal)	Allowing user input to control paths used in filesystem operations	application doesn't sanitize URL directory traversal inputs. When an application improperly cleanses special character sequences in user-supplied filenames	CWE ID 73	M	attacker can gain root access, or get access to files in directories that should be secret	Validate all user-supplied input to ensure that it conforms to the expected format, sanitize input with directory traversal	Permissions on files are correctly set, and all URL inputs are sanitized, and

7(Information Leakage)	disclosure of information that is either regarded as sensitive within the product's own functionality or provides information about the product or its environment that could be useful in an attack	The software generates an error message that includes sensitive information about its environment, users, or associated data (for example)	CWE ID 209	L	information leakage allows attackers to gain information to be used for other malicious attacks	Configure applications and servers to return generic error messages and to suppress stack traces from being displayed to end users.	test error messages to make sure that they don't give away valuable information to users
8(Untrusted Initialization)	an application allows external control of system settings or variables, which can disrupt service or cause an application to behave in unexpected ways	when command line inputs are too long, (i.e. string copy) then this might result in buffer overflow and the execution of arbitrary code	CWE ID 454)	L	an attacker can inject large parameters into the application, causing the input buffer to overflow and any excess input to be executed as code. The application can then execute any extra code	Compartmentalize the application, then treat any input or control outside the trust boundary as potentially hostile.	Don't use the strcpy() function! Instead use strncpy(). Also, the data copied should have an upper bound