|  |
| --- |
| **Category: Input Data** |
| **Threats:**   1. Buffer overflows 2. Cross-site scripting 3. SQL injection 4. Form field manipulation (including hidden fields) 5. Cookie manipulation 6. HTTP header manipulation |
| **Vulnerabilities:**   1. Using non-validated input in the Hypertext Markup Language (HTML) output stream 2. Using non-validated input used to generate SQL queries 3. Relying on client-side validation 4. Using input file names, URLs, or user names for security decisions 5. Using application-only filters for malicious input 6. Looking for known bad patterns of input 7. Trusting data read from databases, file shares, and other network resources 8. Failing to validate input from all sources including cookies, query string parameters, HTTP headers, databases, and network resources |
| **Countermeasures:**   1. Do not trust input 2. Validate input: length, range, format, and type 3. Constrain, reject, and/or sanitize input 4. Encode output |

|  |
| --- |
|  |

|  |
| --- |
| **Category: Sensitive Data** |
| **Threats:**   1. Accessing sensitive data in storage 2. Accessing sensitive data in memory (including process dumps) 3. Network eavesdropping |
| **Vulnerabilities:**   1. Storing secrets when you do not need to 2. Storing secrets in code 3. Storing secrets in clear text 4. Passing sensitive data in clear text over networks |
| **Countermeasures:**   1. Do not store secrets in software 2. Encrypt sensitive data over the network 3. Secure the channel |

|  |
| --- |
| **Category: Cryptography** |
| **Threats:**   1. Loss of decryption keys 2. Encryption cracking |
| **Vulnerabilities:**   1. Using custom cryptography 2. Using the wrong algorithm or a key size that is too small 3. Failing to secure encryption keys 4. Using the same key for a prolonged period of time 5. Distributing keys in an insecure manner |
| **Countermeasures:**   1. Do not develop and use proprietary algorithms. Use platform-provided cryptography 2. Use the **true secure random** method to generate random numbers 3. Avoid custom key management. 4. Periodically change your keys |

|  |
| --- |
| **Category: Session Management** |
| **Threats:**   1. Session hijacking 2. Session replay 3. Man-in-the-middle attacks |
| **Vulnerabilities:**   1. Passing session identifiers over unencrypted channels 2. Permitting prolonged session lifetime 3. Having insecure session state stores 4. Placing session identifiers in query strings |
| **Countermeasures:**   1. Partition site by anonymous, identified, and authenticated users 2. Reduce session timeouts 3. Avoid storing sensitive data in session stores 4. Secure the channel to the session store 5. Authenticate and authorize access to the session store |

|  |
| --- |
| **Category: Auditing and Logging** |
| **Threats:**   1. User denies performing an operation 2. Attacker exploits an application without trace 3. Attacker covers his tracks |
| **Vulnerabilities:**   1. Failing to audit failed logons 2. Failing to secure audit files 3. Failing to audit across application tiers |
| **Countermeasures:**   1. Identify malicious behavior 2. Know your baseline (know what good traffic looks like) 3. Use application instrumentation to expose behavior that can be monitored |

|  |
| --- |
| **Category: Authentication** |
| **Threats:**   1. Network eavesdropping 2. Brute force attacks 3. Dictionary attacks 4. Cookie replay attacks 5. Credential theft |
| **Vulnerabilities:**   1. Using non-validated input in the output stream 2. Using non-validated input used to generate SQL queries 3. Relying on client-side validation 4. Using input file names, URLs, or user names for security decisions 5. Looking for known bad patterns of input 6. Trusting data read from databases, file shares, and other network resources 7. Failing to validate input from all sources including cookies, query string parameters, HTTP headers, databases, and network resources |
| **Countermeasures:**   1. Use strong password policies 2. Do not store credentials 3. Use authentication mechanisms that do not require clear text credentials to be passed over the network 4. Encrypt communication channels to secure authentication tokens 5. Use HTTPS only with forms authentication cookies 6. Separate anonymous from authenticated pages |

|  |
| --- |
| **Category: Authorization** |
| **Threats:**   1. Elevation of privilege 2. Disclosure of confidential data 3. Data tampering |
| **Vulnerabilities:**   1. Relying on a single gatekeeper 2. Failing to lock down system resources against application identities 3. Failing to limit database access to specified stored procedures 4. Using inadequate separation of privileges and permissions |
| **Countermeasures:**   1. Use least privilege accounts 2. Consider granularity of access 3. Enforce separation of privileges 4. Use multiple gatekeepers 5. Secure system resources against system identities |

|  |
| --- |
| **Category: Exception Handling** |
| **Threats:**   1. Revealing sensitive system or application details 2. Denial of service attacks |
| **Vulnerabilities:**   1. Failing to use structured exception handling 2. Revealing too much information to the client |
| **Countermeasures:**   1. Use structured exception handling (by using **try/catch** blocks) 2. Catch and wrap exceptions only if the operation adds value/information 3. Do not reveal sensitive system or application information 4. Do not log private data such as passwords |

|  |
| --- |
| **Category: Configuration Management** |
| **Threats:**   1. Unauthorized access to administration interfaces 2. Unauthorized access to configuration stores 3. Retrieval of clear text configuration secrets 4. Lack of individual accountability 5. Over-privileged process and service accounts |
| **Vulnerabilities:**   1. Using insecure administration interfaces 2. Using insecure configuration stores 3. Storing clear text configuration data 4. Having too many administrators 5. Using over-privileged process accounts and service accounts |
| **Countermeasures:**   1. Use least privileged service accounts 2. Do not store credentials in clear text 3. Use strong authentication and authorization on administrative interfaces 4. Do not use the Local Security Authority (LSA) 5. Avoid storing sensitive information in the Web space 6. Use only local administration |