1. Injection

* Occurs when the data is sent to an interpreter as a part of a query
* Found in :
  + SQL
  + LDAP
  + XPath
  + NoSQL queries
  + OS commands
  + XML parsers
  + SMTP headers
  + Expression languages
  + ORM queries
* Vulnerability:
  + User-data is not validated, or sanitized by the application
  + Non-parameterized calls
* Attack Scenario:
  + SQL call:

String query = “SELECT \* FROM users WHERE custID=’”+ request.getParameter(“id”)+”’”;

* Prevention:
  + Use safe API to provide a parameterized interface
  + Use Object Relational Mapping Tools
  + Use server-side input validation
  + Use SQL controls in the queries

1. Broken Authentication

* Cause of incorrect implementation of application functions associated with authentication and session management.
* Causes to allow attackers compromise passwords, session tokens and others
* Vulnerability:
  + Allows credential stuffing and other automated attacks
  + Allows weak passwords and default passwords
  + Use of weak credential recovery
  + Use of plain text or weakly hashed passwords
  + Ineffective multi-factor authentication
  + Session ID exposed in the URL
  + Invalidation of Session ID are not done properly
* Attack Scenario:
  + When an application session timeouts are not set properly, a user may be able to access the previous account that was logged into the device if the browser was just closed but the account was not logged out.
* Prevention:
  + Employ multi-factor authentication
  + Employ checks of weak-passwords
  + Set password length and complexity policy.
  + Delay failed login attempts progressively and record these attempts that will warn the administrators about the detected malicious activities.

1. Sensitive Data Exposure

* Web applications lack protection not only for the users credential but also the sensitive information in the system.
* Vulnerability:
  + Data are transferred in plain text
  + Sensitive data are stored and backed-up in plain text
  + Use of weak and default crypto keys
  + Encryption is not imposed
* Attack Scenario: When an application encrypts the sensitive information like passwords to store them and is already decrypted once it is retrieved, then the sensitive will not be secured anymore because it is decrypted automatically upon retrieval. This also makes the encryption useless.
* Prevention:
  + Discard sensitive data once it is not to be used anymore
  + Ensure that sensitive data are encrypted
  + Implement encryption with the use of directives
  + With sensitive data responses, disable caching
  + Use strong functions for hashing passwords

1. XML External Entities

* These entities could be used for internal port scanning, internal file sharing, denial of service and remote code execution.
* Vulnerability:
  + Accepts XML from untrusted sources
  + Enabled document type definitions in SOAP bases web services
  + Use of SAML for processing identity within federal security
* Attack Scenario: When the attacker could alter the entity line to a server’s private network or an endless file to impose denial of service
  + <!ENTITY xxe SYSTEM “<https://192.168.1.1/private>” >] >
  + <!ENTITY xxe SYSTEM <file:///dev/random> >] >
* Prevention:
  + Use less complex data formats
  + Upgrade XML processors and libraries
  + Use dependency checkers
  + Use server side input validation
  + Ensure that XML file upload validates incoming XML with validation of XSD

1. Broken Access Control

* Improper implementation of authentication restrictions
* Vulnerability:
  + Evading access control checks
  + Allowing primary keys to be altered
  + Allowing the view and edit option for someone else’s account
  + Promotion of authorization.
  + Manipulation of metadata
  + Misconfiguration of CORS
* Attack Scenario:
  + When an unauthorized user can easily access the admins page by forcing the browsers with the URL target.
    - http://scenario.com/example/getappInfo http://scenario.com/example/admin\_getappInfo
* Prevention:
  + Minimize CORS usage
  + Implement limit requirements by domain models
  + Deactivate directory listing of web server
  + Limit API access
  + Limit controller access
  + Delay failed login attempts progressively and record these attempts that will warn the administrators about the detected malicious activities.