**Code Review Checklist**

**Are HTTP requests accepted?**

* Is every action restricted to only accept the expected request type?
  + Every action should be restricted to only accept the expected request type. If a GET request is expected then reject a POST request. In MVC this can be done by decorating the request with [HttpPost] or [HttpGet] attributes.

**Are actions allowed that change data or state?**

* Is each action that changed data required to be a POST request?
  + Actions that change data or state should always be POST, not GET requests. Including AJAX requests!
* Is every action that changes state or data protected by any an anti-forgery token?
  + Require and validate an CSRF (cross-site request forgery) token on every POST request. Including AJAX requests!
  + For MVC: @Html.AntiforgeryToken() in the Razer view and [ValidateAntiforgeryToken] on the controller action.

**Is any output data from an untrusted sources? (Hint: Your DB is an untrusted source!)**

* Is all output of untrusted data properly escaped?
  + Never output raw strings from an untrusted source into the markup
  + MVC example: Never use Html.Raw() as this disables MVC's encoding of potentially dangerous characters.
    - Html.Raw(JsonConvert.SerializeObject(myModel)) is NOT SAFE. Instead use Html.Raw(Json.Encode(myModel)).

**Is an interpreted language used? (SQL, LDAP, Xpath, NoSQL)**

* Is a paramterized API used?
  + Use a parameterized API. Do NOT use string building techniques for creating queries.

**Do users have more than anonymous access (or is user authentication used)?**

* Is each action protected with an authentication check before any other code is executed?
  + Require authorization and authentication for each action that needs more than anonymous access.
* Do all pages and resources require authentication except for those specifically intended to be public?
* **NOTE**: For a more detailed checklist use section V2 (Authentication) and V3 (Session Management) of [OWASP's Application Security Verification Standard](https://www.owasp.org/index.php/Category:OWASP_Application_Security_Verification_Standard_Project)

**Do privileged users exist?**

* Does the UI limit the display of navigation to authorized function?
* Are server side authorization checks performed on every privileged user function before any other code is executed?
* Are server side checks performed based on trusted information and not solely based on information provide by the attacker?

**Do any users have only partial access to certain types of system data?**

* Are any **direct** references exposed to the user?
  + Does the application verify the user is authorized to access the exact resource requested?
    - When direct object references are used the application must verify that the user is authorized
    - Create object reference maps instead of using direct identitifiers (IDs).
* Are any **indirect** references used?
  + Does the mapping to the direct reference limit the values to those authorized for the current user?

**Are authentication credentials for the user stored?**

* Are the credentials stored using hashing or encryption?
  + Credentials should never be stored in plain text or using weak hashing or encryption algorithms.

**Are cookies used?**

* Are the secure flag and the httponly flag set on every cookie?
  + The secure flag tells the browser not to send the cookie over an insecure connection.
  + The httponly flag prevents the cookie from being accessed via JavaScript.
* Are session cookies used?
  + Do the session cookies only contain randomly generated identifiers?
    - Session cookies should not contain sensitive values.
    - Randomly generated session cookies should not exhibit a predictable pattern.
  + Do the session cookies expire within a reasonable amount of time?
  + Are session IDs properly invalidated on user logout?

**Is error handling used?**

* Is global error handling in place?
  + A global error handler should be used to prevent errors from bubbling out of the application and exposing error messages or stack traces to the user.
  + The global error handler should also be responsible for providing a user-friendly UI in the event of an error.
  + Don't show yellow screens of death in production. Instead, use [Custom Error Pages](https://github.com/NebraskaGlobal/DPLOracle/wiki/Custom-Error-Pages)

**Is this a web site/web application?**

* Is the X-Frame-Options header included on every response?
  + Clickjacking and Cross-Site Request Forgery (CSRF) can be performed by framing the target site. An attack can trick the user into clicking on the link by framing the original page and showing a layer on top of it with dummy buttons.
  + X-Frame-Options: This header works with modern browsers and can be used to prevent framing of the page. Note that is must be an HTTP header, the setting is ignored if it is created as an "http-equiv" meta element within the page.
  + Optional: Framekiller JavaScript - code that prevents the malicious user from framing the page.

**Are redirects or forwards used?**

* If the target URL is included as a parameter, is it validated against a whitelist?
  + If used, don’t involve user parameters in calculating the destination.
  + If destination parameters can’t be avoided, ensure that the supplied value is valid, and authorized for the user.

**Is sensitive data used (For example: passwords, credit card numbers, health records, and personal information)?**

* Is sensitive data encrypted at rest?
* Is sensitive data encrypted in transit?
* Is sensitive data only stored when absolutely necessary?
  + Don’t store sensitive data unnecessarily.
* Is sensitive data discarded as soon as possible?
  + Data you don’t have can’t be stolen.
* Do forms that collect sensitive data disable autocomplete?
* Do pages that contain sensitive data disable caching?
* Is the use of sensitive data compliant with all laws and regulations (e.g. [HIPAA](https://www.hhs.gov/hipaa/for-professionals/index.html), [PCI](https://www.pcisecuritystandards.org/pci_security/), [COPPA](https://www.ftc.gov/enforcement/rules/rulemaking-regulatory-reform-proceedings/childrens-online-privacy-protection-rule), etc.)

**Are frameworks or libraries used?**

* Are the components on the latest stable version?
  + Components should be validated to avoid using components with known vulnerabilities.

**References**

* [OWASP Top Ten Project](https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project)
* [OWASP Application Security Verification Standard Project](https://www.owasp.org/index.php/Category:OWASP_Application_Security_Verification_Standard_Project)
* [OWASP XSS Prevention Cheat Sheet](https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet)
* [OWASP DOM based XSS Prevention Cheat Sheet](https://www.owasp.org/index.php/DOM_based_XSS_Prevention_Cheat_Sheet)