Security related features of the OnlineBanking web application (Dinsmore and Sahai)

Use of SSL – In a production version of the application, it would be deployed using SSL. In the demo environment SSL is impractical due to the cost of a SSL certificate or the requirement to generate a self-signed certificate and require the end user to trust the certificate. The use of SSL ensures that the connection between the client and the server is protected from man-in-the-middle attacks.

Use of Anti-Forgery Token – The anti-forgery token is issued by the server when the page is sent to the client, when the client sends HTTPPost requests, the anti-forgery token is sent along with the request. If the server does not recognize the token, the request is rejected. This ensures that the request is coming from a page that was issued by the server and not by a third party. This is an anti-phishing and anti cross-site request forgery mechanism.

User Registration and Account Creation – Users cannot simply register to use the site. The process calls for the user to open an account and supply an email address that will be used with the account. This prompts the bank to create an account record and associate it with a customer record. When the user registers on the web site, they use the same email address that is associated with the account. In a production environment, this would not be a robust enough solution; at a minimum, two-factor authentication would be required.

Use of Login – Each user must log in to the application; any user who is not logged in is severely restricted in the actions that they can perform.

Secure login storage - The login information is stored in a secure database where the user’s password is encrypted to prevent others, including system administrators, from viewing the password. Using a tested and proven security mechanism such as the login provider reduces our overall risk of creating an incorrect implementation.

Use of Role based authorization – Each controller method (for the banking functions) requires authentication and authorization based on the user’s role(s). Users with the “Customer” role are allowed to perform customer actions, users with the “Operator” role are allowed to perform operator actions.

Use of Encryption in the database – In our initial analysis, we identified an insider attack where the database administrator was able to query the database to get account numbers and other related information. We have encrypted the account number field in every location in the data model to prevent this attack.

Secure Coding – The application was coded and tested to ensure that it implements the functional requirements to keep information secure. For instance, each login is associated with one or more accounts, if the user who is logged in tries to get information about or transfer funds to or from an account that he is not the owner of, the operation will be rejected. This functionality is implemented on the server side to prevent a user from bypassing any client side checking. All of the code is wrapped in try-catch blocks and innocuous exceptions are thrown that do not reveal program structure or details.

IIS ASP.Net Validation – IIS automatically validates incoming data to remove SQL injection strings, additionally, the data model is annotated with tags to indicate other types of validation rules such as the length of incoming strings.