#### **Purpose**

This assignment uses a typical web-based retailing application to show the popularity of using client-side control technologies in practice.  
  
 An online shopping application wants to provide good service to its clients. Some   
basic coding techniques are used to provide the following “needed” supports:

* To enable a *personal touch* feature, cookies are used to piggyback client’s previous purchase information. By doing so, relevant products can be recommended to the client when the client logins onto the website as a returning customer.
* A client’s credentials (e.g., membership information) are also piggybacked using hidden forms to ensure that the client has certain privileged accesses to warrant some promotions.
* A module using client­side JavaScript is used to calculate the final price of the purchased order for a client, so the client can have the transaction done quickly when the order gets submitted to the server.
* The application has implemented a security checkout (e.g., SSL technology) as most online applications do. In other words, when the client does the check out, all client’s credentials (e.g., billing information, shipping address) are indeed encrypted.

#### **Tasks**

Please submit a concise, original post using the following guideline:

* Describe/discuss the potential issues in the above described web application.
* What changes should be made to ensure that the issues you found could be well addressed?

Potential Issues:

Each of the implemented “needed” support functionality has some form of security vulnerability attached. Using cookies to provide information about the client's previous purchase information can be dangerous if not set correctly. A hacker who is snooping on the conversation between the client and server may be able to steal another individual’s purchase history. This may then be used to social engineer their way into having access to the account by being able to recount historical purchase information. Using hidden forms to pass credentials to the server is dangerous because they would become easy to steal. If someone is intercepting packages, there are a lot more chances to steal the login information. Using client-side JavaScript to calculate the final price is just a terrible idea. A hacker may be able to bypass the logic (or change the logic) and inject their own price, effectively stealing the product. Using client side encryption offers little protection to personal information, especially if the hacker is using cross-site scripting.

Improvements:

Fortunately, there are some steps that can be taken to implement this functionality a little safer. Purchase history should only be stored server side. Once the user logs in, it can fetch purchase history for a database, use it to find recommended products, and then return those products to the user. Session authentication should be used to verify user authentication across different site pages, such as Bearer tokens. This would prevent anyone from being able to see plain text credentials if they are snooping. All of the price calculations should be done server side. If you need prices to be calculated client side to display to the user, then those prices should only be used as a display value, not a transaction value. Finally, to enhance the TLS/SSL encryption being done, all billing information, addresses, and other personally identifiable information (PII) should be done server. TLS/SSL encryption should be used once to create the data records, but then it should be referenced by the server anytime the client needs to use it. These are some ways that this online application can improve its security.[1]

References:

1. Stuttard, Dafydd, and Marcus Pinto. *The Web Application Hacker's Handbook : Finding and Exploiting Security Flaws*, John Wiley & Sons, Incorporated, 2011. *ProQuest Ebook Central*, https://ebookcentral.proquest.com/lib/pensu/detail.action?docID=819008.