Commentary Week 4

Ways to secure the web site

One way in which the website could be made secure is by ensuring that the data over https. This is done on the server side and can be configured such that trying to access a web site via the insecure form (http) would automatically redirect to using https instead. The advantage of using https instead of http is that it automatically encrypts the data that is being sent, both from the server to the user and the other way around.

Password length/complexity is another thing to consider in regard to ensure the security of a website, specifically this website; CMSS. This can be easily done in php, using if statements once the entered variables are sent are the form is actioned. You can set what you require the users of your website to input as their password, such as a minimum length of characters, a ‘special’ character, such as a punctation mark etc., as well as upper- and lower-case letters being needed simultaneously.

There are two methods in which data can be sent from the client computer to the server/database; GET and POST. For sending private/confidential information, such as user login details or payment details, then POST should definitely be used instead of GET. This is because when form data is sent via GET, the field names, as well as the data entered by the user, is visible in the client browser’s address bar, constituting a major security risk.

If items were to be sold on the web site, then the web site would have to ensure that the existing methods of encryption and security are more than merely satisfactory, as items being sold on a site means that credit/debit card details will most likely be sent to the server and kept there, meaning security becomes increasingly necessary than if there wasn’t one.

How would you set the session maximum length to 5 minutes?

To set the session maximum length to 5 minutes, you need to manually write a function in PHP to set this, and then to run the script if/when it is needed.

```php
//Start session.
session_start();

//Set the number of minutes
$expireAfter = 5;

//Check to see if the "last user action" session
//variable has been set.
if(isset($_SESSION['last_user_action'])){

    //Figure out how many seconds have passed
    //since the user was last active.
    $secondsInactive = time() - $_SESSION['last_user_action'];

    //Convert the minutes into seconds.
    $expireAfterSeconds = $expireAfter * 60;

    //Check to see if they have been inactive for too long.
    if($secondsInactive >= $expireAfterSeconds){
        //If the user has been inactive for too long,
        // will their destroy session.
        session_unset();
        session_destroy();
    }
}
//Assign the current timestamp as the user's
//newest activity time
$_SESSION['last_user_action'] = time();
```

I have entered some comments in the above code, which should explain their purposes.