A2 Broken Authentication and Session Management Lab

In this lab, we'll hijack the cookie from one session and use it to impersonate a user in another. You'll need a tool that will allow you to read cookies and write or edit cookies. You can read cookies in all developer tools and with Fiddler.

Mounting the attack

1. Open a browser and log in to our website. This will be the victim.
2. Using any cookie viewer, read all the cookies you think would be needed to steal the session. (Hint: you'll probably need the cookies for ASP.NET_SessionId and .ASPXAUTH)
3. Open a different browser. Any other browser or even your same browser in incognito mode will be fine. This will be your attack browser. Navigate to localhost:7777 but do NOT log in.
4. In the attack browser, recreate all those cookies. (Hint: make sure the cookies do not include the port number and make sure they are marked as "session" and "httponly").
5. Refresh the page in the second browser. You should now be logged in as the user from the first browser. Go to "My Account". You should be able to read and even change the user's personal information … credit card info, address info, password, the whole enchilada!

Congratulations (sort of), you've just hijacked the first browser's session.

Hardening the site with IP checking

You can get the client's IP address with Request.UserHostAddress. If the user starts a session from one IP and then we get another request from a completely different IP but with the same session, that should raise a red flag. Let's react when that happens.

6. Find the Login.aspx page. Look for the event that fires when the user logs in.
7. In that event read the IP address from Request.UserHostAddress and store it in a Session object.
8. Now go to a sensitive page like the checkout page. In the button's click event, before we allow the user to actually check out, make sure the current IP address is the same as the one that we stored in Session a few moments before.
9. If they are the same, allow the checkout process. If they are different, put up an error message and abort it.
10. Run and test.

At this point, your attack will still succeed since both browsers are on the same machine in this test but a real attack would be thwarted.

Hardening the site with authentication

We've helped the situation, but realize that a skilled hacker can also manipulate his or her IP address. It isn't easy but it is possible. The foolproof protection is to put all sensitive operations behind authentication. Let's do that for a different page.

Note that the ChangeAccountInfo.aspx page is behind authentication. But if someone steals our session, their browser is trusted by our server. We'll fix that problem by re-asking the user to enter their username and password.

11. Edit the ChangeAccountInfo.aspx page. Add two textboxes, UsernameTextbox and PasswordTextbox. Add a note to the user "For security reasons, please re-enter your username and password.
12. Add this to the top of the code-behind for that page:

```
using System.Web.Security;
```

13. In the button's click event, add lines like this:

```
if (!Membership.ValidateUser(
     UsernameTextbox.Text, PasswordTextbox.Text))
{
    lblFeedback.Text = "Bad username or password. Try again.";
    return;
}
```

You've just added another level of difficulty for the hacker. Even if he/she hijacks the session, he must also know the username and password to change account info.

14. Now run your site and test your changes. Try to change any of the user's information. Enter a bad password. It should prevent you from doing it.

Once you've got authentication on this page, you're protected from session-stealing. You can be finished.