Scummy Russian Hackers

# Cookie usage storage:

To handle concurrent users on the web app, we decided to use cookies to remember user’s login details. The complete list of data stored is as follows: username – any sort of email address – and a session id. Both fields are stored as plain text in the cookies.

To ensure the security goal of integrity, before any confidential code or data is loaded from the server, the username and session id are checked against their stored values in the database. This prevents the user from escalating their privileges by tampering with cookies. In the case where the stored username and session id don’t match, the user is presented with an error. Thus, the data of the person is hidden altogether till the issue with their cookie data is resolved.

Session Id’s are created for a specific user when they log in. The backend server binds their name to their session id. The id itself is a randomly generated string that is a good length to help prevent others from guessing the session id. When a user logs out of the system, the session id is destroyed and is removed from the database.   
  
Session Id’s work better than storing the users type because it prevents the cookies from being tampered with. Our first design stored the user’s type in plain text as a cookie. As insecure as that was, the server would check that the type in the database matched what the type in the cookie. However, a simple attack that knew the username of an admin could break into our site; change the username cookie to the name of the admin and change the user type to “Admin.” Thus, a randomized and personal + unique token assigned at every login was considered more secure and hence we went with this design.

# Web Application Firewall (WAF)

The web application has a firewall implemented. Every piece of data given to the site passes through this filter. Additionally, special data such as usernames and passwords are sent through the attack filter and then through a check for that field.

The way the firewall works with the frontend of the site as described in the following. The firewall runs as a webserver on the same machine as the front end. To distinguish between servers, a unique port number is assigned to it. The reason it is implemented as a standalone unit is that all sites running on the machine can access the one central filter. This makes for higher security as this unit can be thoroughly tested. Also, since there is only one copy, it makes it easier to maintain the security of all sites.

The filter checks the data for any potential attack; it could be XSS or a SQL injection. If the string is parsed and is clear of attacks, the server returns the result “True” back to the frontend, otherwise, a precise message is sent back detailing the error. An attack string is defined as one having illegal substrings. (Substrings are defined in the firewall file.)  
  
Testing was done, it was found that an empty input string would result in the filter producing a 404 error. Bottle’s dynamic address resolution require that the dynamic part have one or more characters. To cope with this situation, I added another route path to ../../../../../../Desktop/Screen%20Shot%202017-10-08%20at%2010.01.40%make sure the empty string case was handled. e,g,   
  
  
The data passed between servers, happens only on a local machine, meaning that unless someone had access to that machine, all data passed between the web app and the filter will remain confidential.

# Interaction between WAF and the backend database

All input to the site is passed through the WAF before being processed by the application. This means that before the backend gets access to any of the data, it has already been checked for threats. This means that all data in the database is assumed to be threat free. Based on this assumption, and the fact that the website has no dynamic content outside of that given by the database, there are no checks on data outside of the input. Simple requests make no interaction with the WAF.   
  
This decision was made because, the attack that could enter the system must occur by providing some input. Since this input is analysed, all data that successfully makes it into the system can be trusted.