1 Augmented Chess: Assignment 7

Group Members

Jacob Holm Mortensen	jmorte14@student.aau.dk
Martin Raunkjær Andersen	marand13@student.aau.dk
Thomas Gwynfryn McCollin	tmccol14@student.aau.dk

1.1 Question 1

> *Which elements of your Web application require protection against threats? Which ones do not require protection? Why?*

During the initial requirements generation we decided that the application would be trust based, in a similar manner to the fklub, and therefore we have no user data protection. The reason for this is that the application holds very little user data and this data is not confidential. However, if the application in the future where to store more user data, such as rank, then user authentication would be required.

User data is not the only security concern of the application. The application maintains a connection between the client side and server side. This connection could potentially be hijacked or submitted to a man-in-the-middle attack. Therefore any request from the server could potentially be harmful to the client.

Lastly the server side resources are also vulnerable to attack. Since most of the applications functionality relies on an active connection to the server, denial of service attacks could seriously affect the application. Thus server resources should not be subject to non-client connections.

1.2 Question 2

> *Describe the techniques that can be used to protect the elements of your Web application that require protection.*

Authentication of users will be performed using passwords. Users will be required to create a password along side their account. This password will be used to validate the user during login. The server side will not store the passwords, but rather a one-way hash and possibly a salt. The client side will hash the password and send the hash to the server, which will compare the hash with the stored hash. In order to validate the server we will use transport layer encryption.

In order to counteract session hijacking, the client and server use Same-Origin policy. This means that the client side will only trust requests from the server side from which the client itself has been served. This does not prevent man-in-the-middle attacks, however. In order to deal with this threat, the application would use transport layer encryption. This last measure will be forgone during the project, but in a real world scenario it is relatively simple to set-up using services such as `letsencrypt.org`.

Finally in order to preserve server resources we will limit resource access to authenticated connections, however the server would still be vulnerable to distributed denial of service attacks, in which each malicious connection is a login attempt. This limits the problem since direct access is disallowed, however server resources could still be exhausted. A better solution would be to use services such as could front.

1.3 Question 3

> *Describe the drawbacks of ensuring the highest possible security for all the elements of your Web application.*

A highly secure application is quite complicated to use. The user would require knowledge about security practices to make sense of the seemingly arbitrary requirements presented by security. An example of this that a highly secure application could require 256 character passwords to ensure a sufficient minimum password complexity. It is unlikely that users can remember passwords this long and it would take a while to type it out every time the application is launched. Thus usability is reduced by increasing security. This is not always the case, technology such as fingerprint or iris readers allows higher security at a lower usability cost, however the technology presents other difficulties, such as privacy concerns about the biometric data.

Furthermore the applications design would change significantly when considering maximum security requirements. Perhaps elements of the application must authenticate and authorise each other, perhaps the user must login to different elements or periodically.

1.4 Question 4

> *Describe one attack on the network, one attack on the user's session and one attack on the browser request that your Web application or its clients could be victim of. In case any of the attack types is not possible for your Web application, explain why.*

A network attack that has already been discussed earlier is a distributed denial of service attack. In such attacks a number of computers, typically infected by a botnet, open connections or request resources from a target system. The idea is to overwhelm the target system with the malicious requests, thus preventing or obstructing valid traffic. This kind of attack reduces a sites availability and is quite difficult to counteract.

Session hijacking or session attacks have also already been mentioned. This is the case when an attacker hijacks an active session, typically an authenticated session. This allows them to bypass authentication, because the user has already authenticated with his/her password. This can be achieved by stealing a valid users cookie and using the information contained within to pose as that user. This can be counteracted using request validation and transport layer encryption.

In a browser request attack an attacker could attempt to cross-site request forgery to pose the server and send the client a forged login request. Thus the user may accidentally supply the attacker with his/her valid password. This can be counteracted using Same-Origin policy and encryption.

1.5 Question 5

> *From the plan given as answer to the question 3 of the assignment 5, which activities have you already performed? how has the Web application been improved after performing these activities?*

In the answer to question 3 of assignment 5 we proposed three different testing techniques to ensure three different criteria. User testing to measure and ensure usability of the application. Cross platform/browser testing to ensure portability. Stress testing to measure system limits and to ensure reliability.

We have conducted portability testing in different browsers throughout development. The group uses a number of different browser when testing to ensure this quality requirement. We have not conducted a formal test of the portability yet. User testing has be conducted informally, in the sense that we have spent time testing the application. However, once the application is more complete a proper user testing will be conducted. Finally we have not conducted stress testing, this is due to that fact that the majority of the work done so far is on the client side. Therefore there is no reason to stress test the server side yet.

1.6 Question 6

> *Continue the development of your Web Application. Report its status.*

The status of the application is as follows. Progress has been made with the move checker and invalid moves are now rejected. Furthermore the previous move is highlighted such that the current player and the previous move is obvious. The final part of the move checking phase now includes a win condition checker. Thus a game consists of of a series of valid moves and the game end when one of the player lose their king. However, the case when a player does not have a valid move is not handled properly yet.

Army management has progress to the point where movement patterns can be modified. This means that army modification is close to completion, the next problem is persistently store the created armies and include them in play.