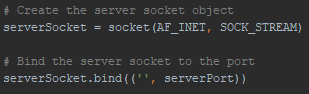
Confidentiality.

In my solution to assignment 1 there are many vulnerabilities where attacker could change or alter code in malicious ways that could cause massive problems for the users on the network. One such vulnerability can be found in the socket connection established between the client and the server. When a messages is sent from one party to another, there is no special encoding or security applied to each message, it is send in plain human readable text.



(Above) Unsecured socket connection (Above) Readable request to the server

(Above) Readable response from the server

This leaves the door open for any attackers to intercept these messages and read exactly what is being sent. For applications such as Naughts and Crosses this is not as detrimental as say a banking system where usernames and passwords to accounts could be intercepted. To alleviate this issue we can incorporate Transport Layer Security, also known as “Secure Sockets Layer”. SSL provides encryption and peer authentication for network sockets for both client side and server-side. It works by establishing a secure encrypted TCP connection. Once established the client will be able to send HTTP requests over the encrypted TCP connection. For a more indepth look at how to implement a SSL socket, more information can be found at this website: <https://docs.python.org/2/library/ssl.html>.

Usability.

For this assignment it was specified that the servers only had to handle one client at a time. This unfortunately opens vulnerabilities when two clients request resources at the same time. If client 1 places an X in the first square on their board, then client 2 places an X in the second square of their board, client 2 will receive client 1’s selection and an X will appear in two spare 1 and 2 of their board. This problem can be solved by incorporating multi-threading. Multi-threading establishes a separate, independent thread for each client meaning that no client can affect the game state of another

Accessibility.

Another vulnerability will come in the form of a Distributed Denial of Service (DDoS) attack. This type of attack is targeted at the server-side using what is called a botnet (a large group of single point controlled computers). The botnet will continuously requests server resources until there are not resources left to handle normal user requests. If this was to happen to a client while they are play my Naughts and Crosses application then will be unable to register their selection. It is difficult to prevent an attack like this as any client can request resources from the server. If a client makes multiple requests at levels similar to a DDoS attack then their IP address could be blocked.