#### **Purpose**

We use a variety of web applications at work, home, and school. Personally, I have to remember dozens of different usernames, passwords, and security questions. As we know, many commercial web applications are vulnerable. This assignment asks you to share your experience and understanding of certain authentication mechanisms in use that might be vulnerable.

#### **Tasks**

Please submit one original post using the following guideline:

* Please recall one example you encountered (you might be still using it now), where the authentication is weak. Please describe the example without disclosing the real name of the web application.
* If you are not sure, how will you test it out? (No real test needed, as it is illegal to run penetration testing against third party applications without permission!)
* If you were the designer of the application, how would you fix it?

**Grading Rubric (Total of 3 points)**

1. An uncovered authentication weakness and its fix? (1.5 points)
2. Quality of your explanation (1.5 points)

Example:

I use one (possibly 2) applications that only rely on username and password for authentication, although MFA is optional. I consider this weak authentication because password leaks happen frequently and if all that is needed to access an important account is the username and password, then hackers can easily gain access to the system. For this particular application, I do have MFA enabled, but it is MFA via SMS. I find this to be the laziest and weakest form of MFA. Hackers can spoof your phone number to bypass the MFA guardrails. This means they would also have the ability to steal your account and lock you out of it.

Toolkits like the ones that we have been using for penetration attacks may be able to identify weakness in the requests that would allow someone to bypass the MFA. It may expose the phone number associated with the account or provide a value that can be modified to bypass MFA altogether.

Fixes:

There are two examples that I can think of that other applications employ to make this more secure. The first is using an authenticator app developed by a trusted third party, like Google Authenticate, instead of using SMS. This would remove the possibility of someone spoofing your phone number to have the SMS authentication routed to them. If the application is a high-risk account, such as banking, the developers should consider a more secure MFA like biometric or hardware authentication. This article has more details on why SMS MFA isn’t as safe as other methods: <https://cyberhoot.com/blog/top-five-risks-from-sms-based-mfa/>. The second idea is using geolocation information to determine where the application is being accessed from. If the client geolocation is from a location different from the usual access location, then an email can be sent to the user asking for permission before allowing the client access to the application. The server can also terminate user sessions if it realizes there are concurrent requests from different locations. This can help limit the time a hacker has with an account.