L2: Discussion 1 - Intercepted Data

#### **Purpose**

In this discussion, I would like you to explain how insecure a web application could be if proper protection is not implemented by the solution provider (the one hosting the web application).

#### **Tasks**

Please submit one original post using the following guideline:

* Provide an example of how your inputs via a web form could be intercepted by attackers/intruders.
* Discuss one specific scenario to articulate the insecurity of web applications when no protection is implemented by the solution provider.
* Present a possible solution to address the insecure scenario.

**Example:** Session Hijacking

**Concept:** One example of how a web application could become insecure is by session hijacking, sometimes referred to as “Cookie Spoofing” or “Cookie Side-Jacking”[1]. To understand what session hijacking is, you will need to understand what a “session” is and why it is important. Essentially, when you login into a website, the server returns and assigns a unique identifier back to your browser. This allows the server to recognize your browser so that when you are navigating between different pages, it does not repeatedly ask you to login with each new page on the website that you visit. Your session will be active for as long as you are logged in or for a preset amount of time[2]. A session can be hijacked by “packet-sniffing.” This is when a program intercepts and monitors the requests and responses between the client and the server[3]. The session identifier could be intercepted and used to spoof your session from another machine.

**Scenario:** A real world example of this happened to a popular technology YouTuber known as Linus Tech Tips. In this instance, a phishing email was sent to someone on their network that was logged into the YouTube account. This email contained an attachment that looked like a PDF, but it was actually executable code that monitors and transmitted locally stored browser information, i.e. session cookies. By stealing the session cookie, they were able to spoof the session and bypass any MFA. The consequence in this case was that the bad actor was able to log into the account, delete all the videos, and start uploading and streaming their own content[4].

**Solution:** In my opinion, the best solution to prevent session spoofing relies on the user being more aware of the networks they are using, especially in public, and not clicking on untrusted links or files. A server side solution is to enforce a time limit on user sessions. This means that the user would have to reauthenticate after the session expiration. This will limit the possibility of a stolen session key from being used to only the time window the server sets for that given session. In the above example, if the employee was more diligent about clicking on files from unknown sources, it would not have happened. From YouTube’s perspective, forcing a re-authentication after a set unit of time may have meant that the actor would have a smaller window to gain access to the system which may have made it more difficult for them.

**References:**

1. <https://blogs.perficient.com/2022/12/27/cookie-spoofing-explained-and-prevented/>
2. <https://www.emsisoft.com/de/blog/44071/what-is-session-hijacking-and-how-do-you-prevent-it/>
3. <https://www.netscout.com/what-is/sniffer>
4. <https://www.neowin.net/news/linus-tech-tips-youtube-channels-were-hacked-due-to-a-session-hijacking-attack/>