CO876 Week 20 Class Worksheet

Budi Arief (b.arief@kent.ac.uk) – Based on materials by Shujun Li
Class Supervisors: Yichao Wang (Y.Wang-301@kent.ac.uk) and Joseph Kearney (J.Kearney@kent.ac.uk)

Friday 17 December 2021

In this week's class, you will work on some exercises beyond user authentication to gain a better understanding of authentication in the wider sense. The first five are easier and mandatory, while the rest are more advanced and optional. The latter can be used for Part 2 of your CW2 assignment if you so wish.

Like in the last week's class, for this week's class exercises, **you are encouraged to use existing third-party libraries and tools as much as you can**, as long as you will still write some source code yourself to get the exercises done. You should use Google and other search engines to get help and look for hints. Work independently first before asking for help.

For all exercises, feel free to show the work you have completed to class supervisors; but if you are confident with your results and do not feel the need to get them checked, that is also fine.

For external websites included in this work sheet, you are recommended to visit them from a more secure and privacy-friendly mode of your web browser as those websites are not necessarily trusted.

Note: This class is the fourth (and the last) class that you will need to include in Part 1 of your CW2 practical report (i.e. the log book). *Please keep any evidence of the work you achieved (e.g. as screenshots or answers to specific exercises), and include them in the practical report that you will submit as part of CW2.*

Exercise 1 – Setting Linux file and folder permissions

Write a script to automatically convert a human-entered Linux permission string (like "rwxr-xr--") into an integer that you whose octal representation is like 754.

You may want to implement it as a simple web page with JavaScript so you can take the permission string from an `<input>` element. If you decide to do so, feel free to make the interface even more user-friendly, e.g., implementing a user interface like the one at https://chmod-calculator.com/.

(Optional) Consider making your script fault tolerate (fixing too short or too long input string, wrong permission flag character at any position, etc.).

Exercise 2 – Investigating a real-world example of role-based access control (RBAC)

Find a real-world example of RBAC and document what roles are defined. This can be your Windows user management system or a website you are involved as an administrator. If you cannot find a good one you are using, search for an online application with RBAC access control (e.g., a web forum, a Wiki, a CMS), install it on your local computer and then examine it.

Exercise 3 – Learning about same-origin policies (SOP) and cross-site scripting (XSS) attacks

Visit this website to play a game to learn about XSS attacks that can compromise the SOPs of a target website: https://xss-game.appspot.com/. Think about how XSS attacks circumvent the SOPs that are supposed to be mandatorily enforced.

Exercise 4 – Playing with a sandbox (a computing one of course, not the one you find on a playground!)

Find a sandbox used by a computing system and examine its security settings and/or any existing restrictions on access control. Think about why such security settings and/or access control policies are needed and how they are enforced. Search for any published vulnerabilities that can compromise the security settings.

For the CW2 report, describe what sandbox you choose and what you learned about it **briefly**.

Hint: Examples include mobile apps, Docker containers, virtual machines, virtual servers (e.g., a virtual server hosted by a web hosting company), and `<iframe>` elements in a web page.

Exercise 5 – Examining a real-world federated identity management (FIM) system

Find a real-world FIM system you've used and examine how technically the system is / may have been implemented. Search for technical documents related to the system to understand more.

For the CW2 report, describe what FIM system you examined and what you learned about it **briefly**.

Hint: To identify the techniques used behind a FIM system, search for its name and examine any technical information you may have access to (e.g., HTML source code returned from a website, source code of the system if published under an open source license).

Exercise 6 (optional) – Using content security policies (CSP) to secure a website

Create a simple dynamic website and think about what you need to do to prevent XSS attacks against the website. Use Google to search for an existing library or class supporting CSP to make your job easier.

Exercise 7 (optional) – Implementing attribute-based access control (ABAC) for a website or a mobile app

Design a simple website or mobile app implementing some form of age-verification techniques (AVTs) to use age as an attribute to control the access to some content of the website/app.

Hint: Check mobile apps for kids on Google Play to see how they use AVTs to unlock advanced settings or solutions (which are reserved for parents or teachers).

Exercise 8 (optional) – Implementing a toy federated identity management (FIM) system

Using any of the techniques introduced in the lecture slides or any other FIM techniques you can find, build a toy FIM system. To demonstrate the working of the system, implement at least one identity provider (IdP), one user, and two service providers (SPs), and demonstrate that the user can use single sign-on (SSO) to log into the IdP once but all SPs without the need to re-logging in repeatedly.