ePortfolio Submission

Please find the ePortfolio for the module under

https://essex.sevale.de/modules/secure-software-dev-cs/eportfolio/.

Module reflection

Before starting the module, I was already familiar with some of the technical areas that it touched on. I had worked on backend systems, written Python and Kotlin code, and used various techniques — although by instinct — to ensure security and reliability. My expectation was that the course would systematise the knowledge I already had as well as introduce the common approaches to writing secure software.

The aspect of the course I found the most useful was the structured introduction to widely used frameworks and classifications like OWASP, STRIDE, and CWE (Microsoft, 2009; OWASP Top 10 team, 2021; The MITRE Corporation, no date). I knew of some of them before but never had a chance to apply them in practice. During the module, I had an opportunity to browse through CWE in more detail and learn about how the vulnerabilities are classified, as well as read the recommendations on preventing them for each of the documented types. I expect that I will return to these resources in the future, even though I am not yet sure how to navigate through CWE efficiently.

Another topic that I appreciated was regular expression safety. I work with regex regularly, and the concept of an “evil regex” was not new to me. However, I have not paid much attention to how the vulnerabilities associated with poorly written regular expressions occur or how to identify and mitigate them. The articles by Goyvaerts (2023) and Weidman (no date) provided insight into how these problems can lead to

denial-of-service scenarios or performance degradation, as well as specific recommendations on how to use regular expressions safely.

Some concepts introduced in the module were genuinely new, like faceted data, as discussed by Schmitz et al. (2016). Although restricting access by authorization levels is not unfamiliar to me, the proposed mechanism involving secure multi-execution was difficult to follow and unlike I had encountered before. I am not sure I have fully internalised the idea, but it encouraged me to think more carefully about sensitive data processing and display in the applications I work on.

The main practical component of the module was the secure school grading system developed during the individual part of the coding assignment. As in the earlier modules, the final implementation diverged from the original design, but the models still provided a solid foundation for the system. The most challenging part of the assignment was implementing the vulnerable mode of the application. While some attack scenarios were straightforward, others, like the injection attack, required further research. Eventually, I decided to add separate logic for payload processing that would allow the attack to happen. This approach worked, but it introduced new challenges: the payloads had to be correctly formatted and debugging injected code proved difficult. Making this part of the system function as intended required more time and experimentation than I initially expected.

The assignment also gave me an opportunity to look more closely at password hashing. Although I was familiar with the topic before, I had never studied it in detail nor worked with bcrypt. I was surprised to discover that the library only produces a single string resistant to rainbow tables attacks (Skanda, Srivatsa and Premananda, 2022; Wolford,

2024). I had not expected that a string of such format could encapsulate versioning, salt, and the actual hash in such a structured and verifiable way. After reviewing how the bcrypt algorithm works, I was impressed by how simple and elegant the solution was, which I did not expect from security-related software.

One of the more reflective moments of the module came during the code demonstration. I was asked which part of the assignment was the most interesting to implement. Answering that question helped me realise that building the vulnerable behaviour — while making sure the application was still functional — was perhaps the least intuitive, most unorthodox, but also fun part of the project. Deliberately disabling validation or adding unsafe code paths went against my habits. The exercise made it clear that designing insecure behaviour in a way that still mirrors realistic mistakes is not trivial. At the same time, it also highlighted how easy it is for some vulnerabilities to occur, even with minimal changes. The Slowloris attack, for example, can be enabled or prevented by a single line in an HTTP server configuration, but the effect is significant: a single attacker can make the service unavailable by exploiting this mistake (Damon *et al.*, 2012).

The team project at the beginning of the module was a more mixed experience. The tutor assigned the team, and we were left to organise the collaboration ourselves. Some members joined promptly and participated actively, others did not respond until the tutor intervened, and some disappeared after a few weeks without any notice. This made it difficult to assess whether we should wait for them to contribute or proceed on our own. It left us frustrated, as we could not plan the work efficiently or divide tasks clearly. Although I am used to working in distributed teams and asynchronous environments,

this situation highlighted how team availability and communication directly affect progress. I would have appreciated more guidelines or check-ins from the university to help resolve such situations sooner.

That said, I did value the opportunity to exchange ideas with others. One suggestion that emerged during our discussions was using HTTP connectivity between the client and the server locally. Although this approach was somewhat expected in the assignment, I had initially not considered it, despite my experience working with networked systems. The discussions were a helpful reminder that even familiar design choices are worth revisiting when presented in a new context.

Overall, the module offered a combination of structured frameworks, exploratory tasks, and implementation challenges that were useful for revisiting familiar practices and exploring new ones. While not all parts of the course introduced new material, the focus on security and the requirement to implement insecure scenarios provided a different and valuable perspective on software development.

References

Damon, E. *et al.* (2012) ‘Hands-on denial of service lab exercises using SlowLoris and RUDY’, in *Proceedings of the 2012 Information Security Curriculum Development Conference*. New York, NY, USA: Association for Computing Machinery (InfoSecCD ’12), pp. 21–29. Available at: https://doi.org/10.1145/2390317.2390321.

Goyvaerts, J. (2023) *Runaway Regular Expressions: Catastrophic Backtracking*. Available at: https://www.regular-expressions.info/catastrophic.html (Accessed: 17 July 2025).

Microsoft (2009) *The STRIDE Threat Model*. Available at: https://learn.microsoft.com/en-us/previous-versions/commerce-server/ee823878(v=cs.20) (Accessed: 21 July 2025).

OWASP Top 10 team (2021) *OWASP Top 10:2021*. Available at: https://owasp.org/Top10/ (Accessed: 11 May 2025).

Schmitz, T. *et al.* (2016) ‘Faceted Dynamic Information Flow via Control and Data Monads’, in *Principles of Security and Trust*. *International Conference on Principles of Security and Trust*, Springer, Berlin, Heidelberg, pp. 3–23. Available at: https://doi.org/10.1007/978-3-662-49635-0_1.

Skanda, C., Srivatsa, B. and Premananda, B.S. (2022) ‘Secure Hashing using BCrypt for Cryptographic Applications’, in *2022 IEEE North Karnataka Subsection Flagship International Conference (NKCon)*. *2022 IEEE North Karnataka Subsection Flagship International Conference (NKCon)*, pp. 1–5. Available at: https://doi.org/10.1109/NKCon56289.2022.10126956.

The MITRE Corporation (no date) *CWE - Common Weakness Enumeration*. Available at: https://cwe.mitre.org/ (Accessed: 21 July 2025).

Weidman, A. (no date) *Regular expression Denial of Service - ReDoS | OWASP Foundation*. Available at: https://owasp.org/www-community/attacks/Regular_expression_Denial_of_Service_-_ReDoS (Accessed: 28 April 2025).

Wolford, B. (2024) *What is a rainbow table attack and how to prevent it?*, *Proton*. Available at: https://proton.me/blog/what-is-rainbow-table-attack (Accessed: 21 July 2025).