**Reflection on the Secure Software Development Module**

*GitHub e-Portfolio URL: [*[*E-Portfolio Link*](https://github.com/mobeen-ali/MSc_SDD_PCOM7E_ePortfolio/tree/main)*]*

**WHAT**

When I enrolled in the Secure Software Development module, the emphasis on *security* immediately resonated. Unlike my earlier bachelor’s studies, which barely touched on this area, the module felt like a gap-closing opportunity to elevate my practice into professional territory.

I approached the Command Line Interface (CLI) application challenge with confidence and ambition. My vision extended beyond the Unit 6 team design. I wanted to develop secure admin tools, integrate context-aware CLI states, and even simulate attacks. However, development coincided with a major role transition at work—I had just become the Artificial Intelligence (AI) Team Lead. Managing client meetings, mentoring junior colleagues, and overseeing model research while studying pushed my endurance.

Despite my enthusiasm, my initial Unit 11 submission received just 41%. It was deeply disappointing and made me realise I had taken the assessment lightly. Rather than retreat, I treated it as a wake-up call. I carefully reviewed the feedback and completely rebuilt the implementation. The restructured system followed a clearer modular design, incorporated enhanced security, and aligned with professional standards.

My first version only addressed four Open Web Application Security Project (OWASP) Top Ten categories: A03, A05, A07, and A10. The revised version covered all ten—from A01 (Broken Access Control) to A10 (Insufficient Logging and Monitoring). Each item was selected based on its real-world relevance and its role in strengthening system defences.

I also overhauled my approach to authentication and encryption. Previously, I stored credentials in JSON files, with little thought to exposure risks. This module introduced me to bcrypt and the cryptography package (Romano and Krüger, 2021), prompting me to redesign the system with proper protections. I implemented Fernet encryption for data at rest, used bcrypt for secure password hashing, and integrated Two-Factor Authentication (2FA) using PyOTP to defend against brute-force and replay attacks.

Tools like flake8 taught me that code clarity is not just about style; it plays a crucial role in building reliable systems. Modular restructuring and adherence to *Python Enhancement Proposal 8 (PEP 8): Style Guide for Python Code* (van Rossum, Warsaw and Coghlan, 2001) made the system more maintainable and testable. I also began creating reusable templates for test-driven development from the start.

Before this module, I had only tested systems manually. Now, I use pytest for structured test suites, including a module that simulates brute-force attacks through `hacker.py`. The clean layout made it easier to isolate behaviour and validate components. Running Bandit revealed hidden issues like hardcoded secrets I had overlooked before.

These habits began to influence my work life too. I introduced flake8 and pytest into internal scripts and ran Bandit on several existing tools. This led to improvements in both security and quality. The mindset of secure development began to spread within my team, not just myself.

**SO WHAT**

Balancing this module with full-time work was difficult. I missed most live seminars due to overlapping client calls and relied on recordings, which made it harder to ask questions in the moment. Still, the Unit 6 group design phase was energising. My teammates were responsive and flexible, and we frequently rescheduled meetings to accommodate one another.

I actively contributed ideas on CLI structure, OWASP threat mitigation, and user role design. We used tools like Mermaid (n.d.) to visualise class relationships clearly. This exposed me to cleaner design techniques that I hadn’t used before.

That collaborative experience shaped my understanding of security risks. I saw how OWASP threats like A03 (Injection) and A10 (Logging) could be directly mitigated with Click decorators and login attempt logging. While these concepts were reflected in the team design, my personal implementation extended and tested them thoroughly.

The biggest shift for me was in how I understand testing and code quality. I no longer view clean code as just aesthetic, but as essential for collaboration, maintenance, and reliability. I began applying simplified testing approaches at work, which helped improve our team’s development practices.

Revisiting tutor feedback, reevaluating design decisions, and iterating systematically helped me develop better instincts for aligning design with implementation goals. I now invest more time at the beginning to align expected outcomes with actual behaviour—skills I didn’t have before.

**NOW WHAT**

This module has reshaped my development approach. Encryption, testing, and modular design are no longer afterthoughts, they’re my defaults. Every script I write is structured, secure, and testable. Configuration files are encrypted, and each new project includes linter checks and test suites by default.

Looking ahead, I plan to collaborate more closely with peers in future modules. While my group was open and supportive, I sometimes hesitated to engage early. Next time, I want to communicate more consistently from the beginning to enrich both the learning and final outcomes.

This module also renewed my motivation for the MSc. Initially, I was nervous about whether I could keep up with security-focused topics while handling a new leadership role. But seeing the direct impact of these practices in both academic and professional contexts has been inspiring. Unit 10’s content on faceted data and secure flows sparked new ideas, such as using decorators to enforce data-access layers. These techniques are directly applicable to my current projects.

Finally, compiling this portfolio taught me the value of structured documentation. I developed submission checklists, artefact indexes, and navigation guides to ensure clarity. These habits have now become part of how I manage and communicate updates to stakeholders and developers in my leadership role.

**Conclusion**

Secure software development is not just a checklist; it is a mindset. This module taught me to integrate security into every phase, from design and implementation to testing and review. The lessons I learned, such as applying OWASP principles, using proper encryption, and adopting test-driven practices, are now embedded in how I work and lead.

I began this module with curiosity and am leaving it with conviction and confidence.

**References**

OWASP Foundation (2021) *OWASP Top Ten: The Ten Most Critical Web Application Security Risks*. Available at: <https://owasp.org/Top10> (Accessed: 4 June 2025).

Romano, F. and Krüger, H. (2021) *Learn Python Programming*. 4th edn. Birmingham: Packt Publishing.

Schmitz, T., Probst, F., Ulbrich, S. and Holzmann, C. (2016) ‘Preventing data leakage with faceted data’, *Proceedings of the 21st ACM on Symposium on Access Control Models and Technologies*, pp. 65–75. Available at: <https://doi.org/10.1145/2933446.2933450> (Accessed: 14 July 2025).

van Rossum, G., Warsaw, B. and Coghlan, N. (2001) *PEP 8 – Style Guide for Python Code*. Python Software Foundation. Available at: <https://peps.python.org/pep-0008/> (Accessed: 14 July 2025).