Reflective Piece Secured Software Development

This module focused on Secured Software Design. There were 6 Units in the Module. Dr Cathryn People was the Tutor. The unit started with a collaborative learning discussion forum followed by other practical activities and Assignments which were all centered around Security in Software Development.

Teams were set up for group activity and assignments. I was part of the group 1 named as Team Confidentiality and comprised of Hamad Ahamd, Tebogo Sodaba, Nkosana Mlambo and Myself. Nkosana Mlambo however, dropped the course as a result of an emergency which now left only 3 members.

We had several meetings using communication channels such as Whatsapp, Google meet and Microsoft Team. We agreed on preparing an online Google shared folder granting access to all Team members so that documents can be shared and Edited online.

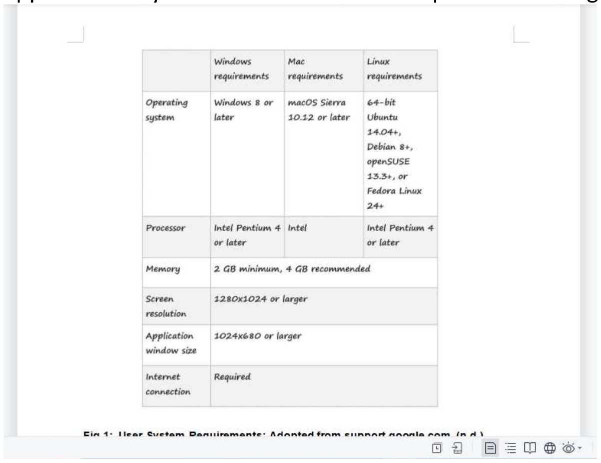
I worked with basically the same team mates in my previous modules but this time, there was a change. I was nevertheless excited to know what my other team mates would bring on board. The good thing about this course as I have observed, is the way it brings people from different part of the world together to share ideas and Knowledge.

The Collaborative learning discussion, is one other interesting part of the study. The discussion in this module was basically on the UML flow chart. It was interesting to see different flow charts designed by my mates showing the different coding weaknesses identified by OWASP, and the steps that led to the weakness occurring. Although I designed a flow chart showing the weakness identified in Cross-Site Scripting (XSS), a post made by my colleague Jonathan Ashmore(Ashmore, 2022) on weak password recovery mechanism for forgotten password caught my attention it appears to be a common practice I have noticed in many of the online accounts I have used in the past. It is quite worrisome that this weakness has not been identified by most online service providers.

The lecture case in this module captured several important topics which include, among others, Architecture of software design, UML flowchart, OWAPS Identified 10 top software security weaknesses, The software development Life Cycle, Waterfall Model, Agile Model, DevOps, TOGAF, Design patterns, ISO/IEC Standard 27000, **Programming Languages,** compilers, Software testing, blockchain, cyber physical systems, microservice systems, The GDPR and Application programming Interface(Essex online Lecture cast, 2021). These are crucial topics that have touched several aspects of security in Software Design. I understand that it is not just about knowing the coding weaknesses or security challenges in software but also, the ability to understand how these softwares are developed from the coding, to the deployment. This will enable one to have an in-dept knowledge on the entire system, so that its becomes easy to nip a security challenge in the bud. These topics have covered the foundation as well as advance aspects of secured software development

There were seminars presented on Scrum Security Review, Cryptography and Programming Languages. I realise that the idea of developing software in sprints (time-boxed period when a scrum team works to complete a set amount of work) is a more effective way of dealing with security issues in a consistent manner. Cryptography is also essential as a requirement to meet the General Data Protection Regulation GDPR (GDPR, 2019). Although I am new to programming, I have also observed that the Python programming language, apart from being more interactive with its almost similar syntax to English language, its less complex nature makes it easier to identify software lapses than the other programming languages.

The Assignments in this module are in two parts and submitted as a group work. As earlier stated, I was in Group 1. The first part was submitted in the third week while the second part was submitted in the sixth week. The first part introduced the major requirement of the project which involves the deployment of a web app for NASA, while the second part involves coding and development of the application. My contributions for the first part of the Assignment can be seen below



	Windows requirements	Mac requirements	Linux requirements
Operating system	Windows 8 or later	macOS Sierra 10.12 or later	64-bit Ubuntu 14.04+, Debian 8+, openSUSE 13.3+, or Fedora Linux 24+
Processor	Intel Pentium 4 or later	Intel	Intel Pentium 4 or later
Memory	2 GB minimum, 4 GB recommended		
Screen resolution	1280x1024 or larger		
Application window size	1024x680 or larger		
Internet connection	Required		

Fig 1: User System Requirements: Adopted from support.google.com (n.d.)

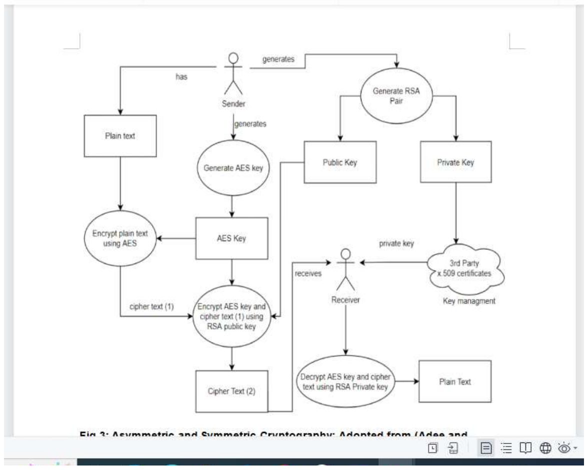
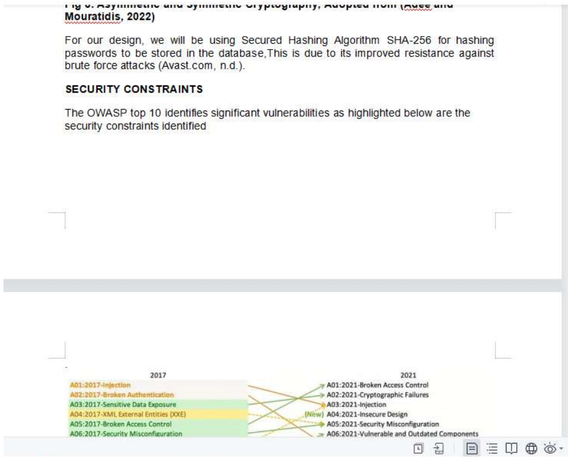


Fig 3: Asymmetric and Symmetric Cryptography; Adopted from (Adee and Mouratidis, 2022)



For our design, we will be using Secured Hashing Algorithm SHA-256 for hashing passwords to be stored in the database,This is due to its improved resistance against brute force attacks (Avast.com, n.d.).

SECURITY CONSTRAINTS

The OWASP top 10 identifies significant vulnerabilities as highlighted below are the security constraints identified

2017
A01:2017-Injection
A02:2017-Broken Authentication
A03:2017-Sensitive Data Exposure
A04:2017-XML External Entities (XXE)
A05:2017-Broken Access Control
A06:2017-Security Misconfiguration
2021
A01:2021-Broken Access Control
A02:2021-Cryptographic Failures
A03:2021-Injection
(New) A04:2021-Insecure Design
A05:2021-Security Misconfiguration
A06:2021-Vulnerable and Outdated Components

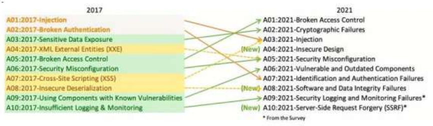
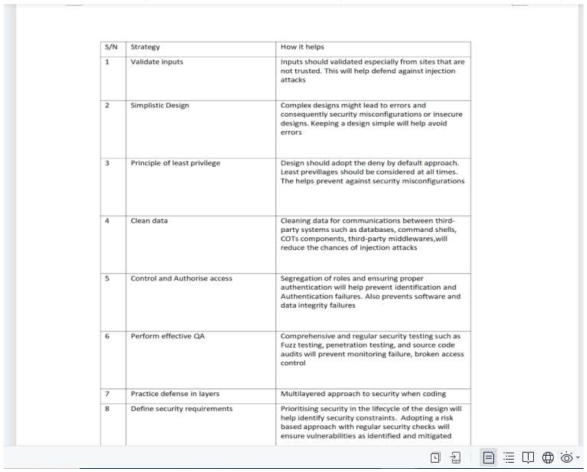
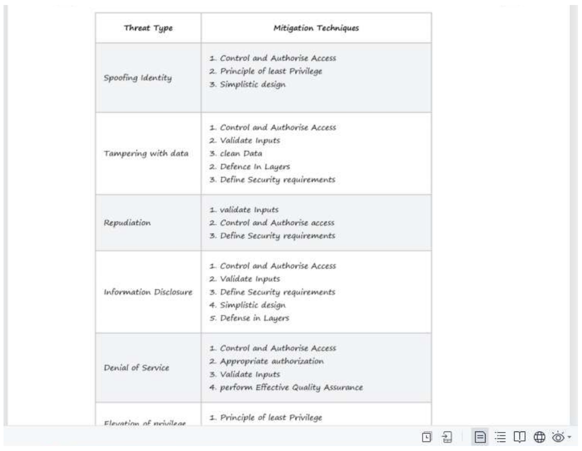

2017
A01:2017-Injection
A02:2017-Broken Authentication
A03:2017-Sensitive Data Exposure
A04:2017-XML External Entities (XXE)
A05:2017-Broken Access Control
A06:2017-Security Misconfiguration
A07:2017-Cross-Site Scripting (XSS)
A08:2017-Insecure Deserialization
A09:2017-Using Components with Known Vulnerabilities
A10:2017-Insufficient Logging & Monitoring
2021
A01:2021-Broken Access Control
A02:2021-Cryptographic Failures
A03:2021-Injection
(New) A04:2021-Insecure Design
A05:2021-Security Misconfiguration
A06:2021-Vulnerable and Outdated Components
A07:2021-Identification and Authentication Failures
(New) A08:2021-Software and Data Integrity Failures
A09:2021-Security Logging and Monitoring Failures*
(New) A10:2021-Server-Side Request Forgery (SSRF)*
* From the Survey

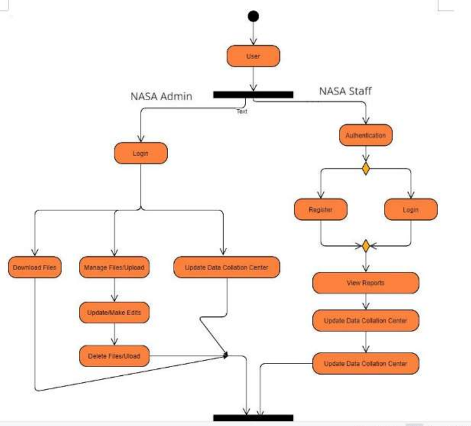
Fig 4 OWASP TOP 10. Adopted from OWASP TOP 10 (OWASP (2021)

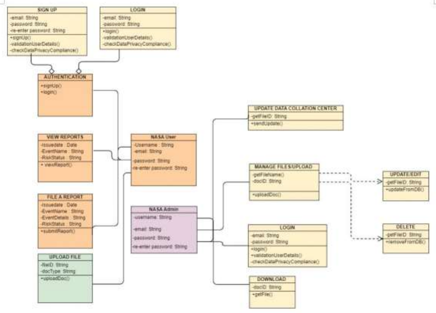


S/N	Strategy	How it helps
1	Validate inputs	Inputs should validated especially from sites that are not trusted. This will help defend against injection attacks
2	Simplistic Design	Complex designs might lead to errors and consequently security misconfigurations or insecure designs. Keeping a design simple will help avoid errors
3	Principle of least privilege	Design should adopt the deny by default approach. Least previllages should be considered at all times. The helps prevent against security misconfigurations
4	Clean data	Cleaning data for communications between third-party systems such as databases, command shells, COTs components, third-party middlewares,will reduce the chances of injection attacks
5	Control and Authorise access	Segregation of roles and ensuring proper authentication will help prevent identification and Authentication failures. Also prevents software and data integrity failures
6	Perform effective QA	Comprehensive and regular security testing such as Fuzz testing, penetration testing, and source code audits will prevent monitoring failure, broken access control
7	Practice defense in layers	Multilayered approach to security when coding
8	Define security requirements	Prioritising security in the lifecycle of the design will help identify security constraints. Adopting a risk based approach with regular security checks will ensure vulnerabilities as identified and mitigated



Threat Type	Mitigation Techniques
Spoofing Identity	1. Control and Authorise Access 2. Principle of least Privilege 3. Simplistic design
Tampering with data	1. Control and Authorise Access 2. Validate Inputs 3. clean Data 2. Defence In Layers 3. Define Security requirements
Repudiation	1. validate Inputs 2. Control and Authorise access 3. Define Security requirements
Information Disclosure	1. Control and Authorise Access 2. Validate Inputs 3. Define Security requirements 4. Simplistic design 5. Defense in Layers
Denial of Service	1. Control and Authorise Access 2. Appropriate authorization 3. Validate Inputs 4. perform Effective Quality Assurance
Elevation of privilege	1. Principle of least Privilege





Note: to illustrate compliance with GDPR, the + shows public attributes while - private attributes

Fig 9: Class Diagram

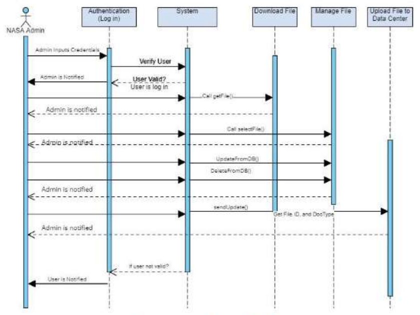


Fig 10: Sequence Diagram for Admin

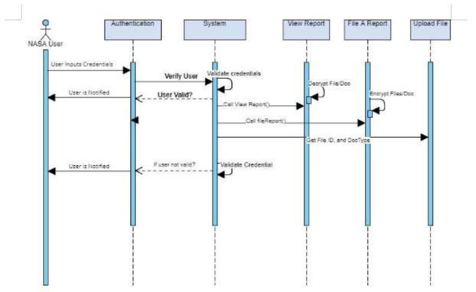


Fig 11: Sequence Diagram for Users

The Web application Architecture Using the Django Framework

Fig 11: Sequence Diagram for Users

The Web application Architecture Using the Django Framework

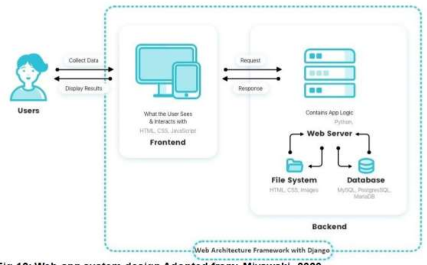


Fig 12: Web app system design Adopted from: Miyawaki, 2020

In the second part which is the coding output, I was able to implement limit to multiple failed loging attempt, Encryption and the interactive features (HTML, JavaScript and CSS) of the web application

This module has exposed me to secured software development. I believe that as a Cyber security specialist, it is important to understand the different coding practices and ways to avoid vulnerabilities. This module has given me the required skill and expertise to manage Cyber crime from the software perspective

Reference

Ashmore, J (2022). Initial Post- weak password recovery mechanism for forgotten password [online] Available from: https://www.my-course.co.uk/mod/hsuforum/discuss.php?d=320662 [Accessed 17 September 2022].

Essex online Lecture cast, 2021. Future Trends. Available from:https://www.my-course.co.uk/Computing/Cyber%20Security/SSD/SSD%20Lecturecast%203/content/index.html#/lessons/zEsHTXkYlFMnESVSiN0uOrdUNKWnBV78. [Accessed 16 September 2022]

General Data Protection Regulation (GDPR). (2019). Encryption | General Data Protection Regulation (GDPR). [online] Available at: https://gdpr-info.eu/issues/encryption/.