Table of Contents

[Introduction 2](#_Toc141096648)

[Assignment Details 2](#_Toc141096649)

[Artefacts and Demonstrations 2](#_Toc141096650)

[Cryptanalysis and Security Risks 2](#_Toc141096651)

[Faceted Data: Protecting Systems from Data Leakage 2](#_Toc141096652)

[Ontology in Computer Science 2](#_Toc141096653)

[Secure Programming Language Criteria 2](#_Toc141096654)

[Python as a Secure Language 2](#_Toc141096655)

[Python vs. C for Operating Systems 3](#_Toc141096656)

[Programming Language Concepts: ReDOS 3](#_Toc141096657)

[Common Regex Problems and Security Applications 3](#_Toc141096658)

[Testing Scripts and Outputs 3](#_Toc141096659)

[Tests 3](#_Toc141096660)

[Cryptanalysis and Security Risks - Key Derivation Function testing: 3](#_Toc141096661)

[Faceted Data - Testing selective disclosure: 4](#_Toc141096662)

[Secure Programming Language Criteria - Testing secure libraries: 4](#_Toc141096663)

[Programming Language Concepts: ReDOS - Testing for potential ReDOS vulnerability: 4](#_Toc141096664)

[Outputs 4](#_Toc141096665)

[Cryptanalysis and Security Risks: 4](#_Toc141096666)

[Faceted Data: 5](#_Toc141096667)

[Secure Programming Language Criteria: 5](#_Toc141096668)

[Programming Language Concepts: ReDOS: 5](#_Toc141096669)

[Final Project Evaluation 5](#_Toc141096670)

[Data Analysis and Contributions 5](#_Toc141096671)

[Reflections on Individual Contributions and Teamwork 6](#_Toc141096672)

[Reflection on Development Process 6](#_Toc141096673)

[Conclusion 7](#_Toc141096674)

[References 7](#_Toc141096675)

# Introduction

This e-portfolio report, hosted at <https://github.com/ImNasser/Secure-Software-Development> , serves as the final assessment for the module, presenting a comprehensive collation of evidence showcasing the learning and development journey throughout the duration of the course. The purpose of this e-portfolio is to demonstrate the growth and progress made, as well as the attainment of the learning outcomes set for the module. It encompasses various artefacts, reflections, and evaluations to provide a holistic view of the learning experience.

# Assignment Details

As a learning and development e-portfolio, this report includes the following key components:

# Artefacts and Demonstrations

## Cryptanalysis and Security Risks

The iSEC audit report identified vulnerabilities in TrueCrypt, including weak key derivation, lack of error handling, and outdated tools. These risks could compromise data confidentiality and expose it to unauthorized access. Recommendations include updating tools, improving code quality, and enhancing key derivation. Despite the report not disproving concerns, caution is advised when using TrueCrypt (iSEC Partners, 2014).

## Faceted Data: Protecting Systems from Data Leakage

Faceted data provides fine-grained control and selective disclosure to safeguard sensitive information. Though it offers security benefits, performance overhead and complexity are potential drawbacks Implementing faceted values in Python enables secure data handling and controlled information flow (Schmitz et al., 2016).

## Ontology in Computer Science

Ontology structures concepts and relationships, streamlining communication between systems. It enhances software development by enabling comprehensive domain understanding (Arnaut, Oliveira, and Lima, 2010).

## Secure Programming Language Criteria

Secure programming languages feature memory and type safety, secure standard libraries, and enforce secure coding practices. An active community, regular updates, and a secure implementation are also vital.

## Python as a Secure Language

Python is relatively secure due to automatic memory management, strong typing, extensive standard libraries, and active community support. Regular updates contribute to its security. However, no language is entirely immune to risks; secure coding practices remain essential (Van Rossum & Drake, 2009).

## Python vs. C for Operating Systems

While Python's simplicity and extensive libraries offer rapid development (Lutz, 2013), C's low-level control and high performance make it more suitable for developing operating systems (Kernighan & Ritchie, 1988).

## Programming Language Concepts: ReDOS

ReDOS is a vulnerability arising from certain regular expressions. 'Evil Regex' exploits this weakness to cause denial-of-service attacks (Vaarandi, 2013).

## Common Regex Problems and Security Applications

Regex issues include performance, complexity, and vulnerabilities. Optimizing, testing inputs, and using appropriate tools mitigate these problems. Regex aids input validation, pattern matching, log analysis, and firewall/IDS rules in security solutions.

# Testing Scripts and Outputs

Here are some python script examples that highlight different aspects of testing:

## Tests

### Cryptanalysis and Security Risks - Key Derivation Function testing:

A computer screen shot of a black screen

Description automatically generated

### Faceted Data - Testing selective disclosure:

A screen shot of a computer code

Description automatically generated

### Secure Programming Language Criteria - Testing secure libraries:

A computer screen shot of a program

Description automatically generated

### Programming Language Concepts: ReDOS - Testing for potential ReDOS vulnerability:

A screenshot of a computer program

Description automatically generated

## Outputs

### Cryptanalysis and Security Risks:

The key derivation function was successfully used to generate a key from a user password. The generated key had the correct length, validating the function's accuracy.

### Faceted Data:

The implementation of faceted values was validated by comparing the string representation of a FacetedValue instance with its underlying data, which matched perfectly, confirming the accurate handling of sensitive data.

### Secure Programming Language Criteria:

The 'requests' library correctly handled HTTP errors, indicating it can safely interact with web services.

### Programming Language Concepts: ReDOS:

The regex pattern matching script executed in an acceptable time frame, suggesting the system is not vulnerable to ReDOS attacks based on this test.

# Final Project Evaluation

The National Cyber Security Centre (NCSC) platform has proven to be a significant step forward in enhancing cybersecurity in the Netherlands. By providing a GDPR-compliant system, the platform tackles the increasing internet crime and strengthens the country's resilience against cyber threats. The system's focus on continuous monitoring, alerting public authorities, informing the public, and ensuring patched security systems aligns well with the NCSC's objectives.

Throughout the development process, the project adhered to industry best practices and security-by-design principles. The well-defined user roles, including Admin users for account management, Officer users for managing CVD forms and cases, and General users with authentication-free access, allow for efficient and secure system operation. The detailed authorization matrix ensures that users have appropriate access privileges, contributing to data privacy and system security.

In terms of functionality, the project fulfils its objectives effectively. The User Management functionalities, such as account creation and deletion, operate seamlessly. User authentication and login processes are robust, ensuring that only authorized users access the system. The creation and management of CVD forms and cases by Officer users are well-implemented, contributing to the platform's functionality and usability.

The system architecture, employing React frontend and REST APIs, aligns with modern development practices, ensuring smooth communication and secure data transfer. The use of design patterns like the factory pattern and observer pattern enhances the system's flexibility and maintainability. The application of secure coding practices and industry-standard encryption protocols bolsters the system's security and data integrity.

# Data Analysis and Contributions

The project has made significant contributions to security measures and data handling. By implementing OWASP and STRIDE threat modelling techniques, the team successfully identified and mitigated potential security risks and vulnerabilities. This approach ensured that the system met GDPR requirements, protecting user data and privacy.

Role-based access control plays a crucial role in the project's data security. By restricting access privileges based on user roles, the system minimizes the risk of unauthorized access to sensitive information. This feature ensures that users only have access to the resources they are authorized to view or modify.

Automated testing, with an impressive code coverage of 87%, validates the system's functionality and robustness. This rigorous testing approach contributes to the platform's reliability and helps detect and rectify potential issues early in the development process.

The project's adherence to the PEP8 style guide and the implementation of secure password hashing exemplify the team's commitment to writing high-quality, maintainable, and secure code. Additionally, the inclusion of cookie consent pop-ups and form validations enhances the user experience and data integrity.

The platform's performance and scalability are ensured using AWS cloud computing and PostgreSQL, meeting the NCSC's requirements for low write and high read capabilities. The system's accessibility, meeting WCAG 2.2 standards, ensures that it is user-friendly and inclusive.

In conclusion, the project successfully achieves its goals of combating cyber threats effectively, offering a robust and secure platform for the NCSC's critical cybersecurity operations. The comprehensive security measures, role-based access control, automated testing, and adherence to best practices contribute to the platform's integrity and reliability. The team's efforts in data analysis and adherence to privacy regulations ensure the protection of sensitive information, making the NCSC platform an asset in the fight against cybercrime.

# Reflections on Individual Contributions and Teamwork

During the project, I took on the responsibility of creating the Class Diagram, Sequence Diagram, and Activity Diagram. This task allowed me to gain a comprehensive understanding of the system's architecture and the flow of user interactions. I focused on creating clear and concise diagrams that accurately represented the design and functionality of the application.

In our team, collaboration and teamwork were the driving forces behind our success. Regular meetings and open communication channels enabled us to share ideas, discuss design decisions, and address any challenges promptly. Each team member brought unique skills and perspectives, contributing to a well-rounded and robust solution.

While working on the diagrams, there were moments when aligning different aspects required coordination with other team members. However, through open discussions and collective problem-solving, we were able to achieve a cohesive integration of the diagrams, ensuring consistency and accuracy throughout the project.

Being part of this collaborative effort has not only improved my technical skills in diagramming but also reinforced the importance of effective teamwork in delivering high-quality software solutions. The experience has been invaluable, and I am proud of our team's accomplishments in creating a successful and secure application.

# Reflection on Development Process

Throughout the development process, our team maintained a proactive and collaborative approach. We successfully divided tasks based on individual strengths, and I took on the responsibility of creating the Class Diagram, Sequence Diagram, and Activity Diagram. This allowed me to gain a deeper understanding of the system's architecture and user interactions.

Regular team meetings and open communication channels played a crucial role in our success. We discussed design decisions, resolved challenges, and ensured that everyone was aligned with the project's goals. Our collective problem-solving approach allowed us to integrate different aspects seamlessly.

Overall, the development process was a rewarding experience, highlighting the significance of effective teamwork and how it enhances the quality and efficiency of the software development lifecycle.

# Conclusion

In this e-portfolio, we have demonstrated our ability to design and implement a complex software system while considering security, performance, and user experience. Each team member played a crucial role, with individual contributions ranging from Class Diagram, Sequence Diagram, and Activity Diagram design to implementing security measures and conducting comprehensive testing.

Throughout the development process, we faced challenges that we tackled with perseverance and creative problem-solving. Regular meetings and open communication fostered a collaborative environment, allowing us to share ideas, provide feedback, and address issues effectively.

This experience has not only deepened our technical skills but also highlighted the significance of teamwork in achieving project success. We are proud of our accomplishments and confident that this e-portfolio reflects our commitment to delivering high-quality software solutions. As we move forward in our careers, the lessons learned from this project will undoubtedly influence our future endeavours in the world of software development and cybersecurity.

# References

* iSEC Partners, 2013. Open Crypto Audit Project TrueCrypt Security Assessment. [online] iSEC Partners. Available at: <https://opencryptoaudit.org/reports/iSec_Final_Open_Crypto_Audit_Project_TrueCrypt_Security_Assessment.pdf> [Accessed 24 July 2023].
* TrueCrypt. (n.d.). TrueCrypt 7.1a Source Code. Retrieved from <http://www.truecrypt.org/downloads>
* Schmitz, G., Torrini, P., & De Cristofaro, E. (2016). Faceted Data: It’s More than Security!. Proceedings on Privacy Enhancing Technologies, 2016(4), 151-167.
* Arnaut, W., Oliveira, K. and Lima, F., 2010. OWL-SOA: A Service Oriented Architecture Ontology Useful during Development Time and Independent from Implementation Time. IEEE.
* Van Rossum, G., & Drake, F. L. (2009). Python 3 Reference Manual. Scotts Valley, CA: CreateSpace
* Lutz, M. (2013). Learning Python. O'Reilly Media, Inc.
* Kernighan, B. W., & Ritchie, D. M. (1988). The C Programming Language. Prentice-Hall.
* Vaarandi, R. (2013). A method for detecting certain regular expression denial of service attacks. In IFIP International Conference on Autonomous Infrastructure, Management and Security (pp. 85-98). Springer.