Kenyhacta Busby

CS 499

11/22/2024

Enhancement One: Software design and engineering

**Briefly describe the artifact. What is it? When was it created?**

The artifact is a security-focused project created in C++ as part of the CS 405 course. It involves creating uniform implementation guidelines, presenting a security policy guide for Green Pace, and ensuring that the development team adheres to best practices. It includes writing unit tests to identify vulnerabilities and addressing risks using 10 guiding security principles.

**Justify the inclusion of the artifact in your ePortfolio. Why did you select this item? What specific components of the artifact showcase your skills and abilities in software development? How was the artifact improved?**

I chose this artifact because it demonstrates my ability to apply security principles in software development and adapt code to meet modern standards. It highlights critical skills, such as secure coding practices, risk assessment, and the application of adversarial thinking to protect against vulnerabilities. These abilities are key for a career in cybersecurity and align well with industry expectations. Key components include the implementation of unit tests in C++ to detect vulnerabilities, the development of a threat matrix, and the use of the Triple-A framework for enhanced security. The planned improvements, such as migrating the C++ codebase to Python, adding retry limits, enhancing input validation, and implementing SQL injection prevention, showcase my ability to refine and enhance software for security and usability.

**Did you meet the course outcomes you planned to meet with this enhancement in Module One? Do you have any updates to your outcome-coverage plans?**

My capacity to create safe, effective, and thoroughly documented solutions is demonstrated by this enhancement, which shows a strong connection with the course outcomes. The design of my enhancement reflects the first outcome, which involves using techniques to create collaborative environments. Program readability and maintainability are improved by splitting the program into smaller, independent functions, which facilitates team collaboration. Clear reasoning and thorough comments make it easier for stakeholders or other developers to understand the function of each component, promoting collaboration and smooth integration. Through its user-facing messages and logging system, the enhancement guarantees understanding and accessibility for the second outcome, which is creating and delivering professional-quality communication. While the use of a logging framework captures important events, errors, and performance information, user messages offer instant and clear feedback. This expert approach guarantees that the person reading the program can efficiently track problems, analyze system performance, and understand the behavior of the code that they're working with. Effective design and careful problem-solving lead to the third outcome, which is the design and evaluation of computing solutions using algorithmic principles. By ensuring that only accurate and safe data is processed, input validation lowers the possibility of mistakes or malicious attacks. The memory allocation function effectively simulates resource management, and the retry mechanism maximizes program flow by limiting invalid attempts. Every function optimizes user experience and performance by adhering to algorithmic standards.

The program's use of contemporary Python libraries and best practices demonstrates the fourth outcome, which is the application of well-founded and creative techniques, skills, and tools. Performance tracking, error handling, and audit trails are all made possible by the logging system and are useful tools in actual software development. A further layer of accountability is added by integrating session authentication using distinct tokens, which guarantee that every program instance can be safely tracked. These methods show my ability to provide solutions that add value while addressing industry-specific objectives and are in line with industry standards. Lastly, the program's design places a strong emphasis on the fifth outcome, which is cultivating a security mindset that anticipates exploitation. By identifying unsafe characters and rejecting invalid inputs, the input validation function proactively guards against SQL injection. The retry mechanism lowers security risks by limiting repeated invalid attempts, which limits potential abuse. The program shows a strong focus on addressing vulnerabilities, guaranteeing data security, and preserving system integrity by combining these features with error logging and session-based monitoring.

**Reflect on the process of enhancing and modifying the artifact. What did you learn as you were creating it and improving it? What challenges did you face?**

Enhancing and modifying the artifact taught me a lot about working with legacy code and transitioning it to a more modern and user-friendly language like Python. I learned how critical it is to keep your projects organized because you never know when you might need to refer back to them. One of the biggest challenges I faced was adapting C++ features like memory management and buffer handling to Python, which required me to really dive into the differences between the two languages. I also found it tricky to balance adding new security features, like retry limits and SQL injection prevention, without overcomplicating the code, but it was a great learning experience overall.