Saxton Dees

Individual Reflection

SecureSim was a multifaceted project designed to simulate a wastewater treatment process while testing cybersecurity threats and defense strategies. Our goal was to build a system that was complex enough to be engaging yet manageable enough to focus on security insights rather than just simulation mechanics. My role centered on refining core simulation components, improving logic, and implementing attack and defense mechanisms. What stood out most to me was seeing how seemingly simple attacks, like a DoS or a replay, could severely impact system availability and reliability.

As a daily Python user, I was comfortable with the language but had limited experience with some of the libraries we used. My contributions started with helping design the simulation structure. We first outlined the core components on a whiteboard before translating them into JSON. I then worked on refining the visual layout generated by the Python library, ensuring that images and labels were correctly spaced and readable. Additionally, I improved system logic by modifying pumps and tanks to log overflow events and prevent pumping into full tanks, enhancing realism and preventing unintended behaviors.

Beyond the core simulation, I implemented key security mechanisms. For the attack side, I developed the DoS attack, which demonstrated how rapid message bursts could cripple system communication. To counteract this, I created a rate-limiting defense that mitigated excessive write attempts, preventing attackers from destabilizing the system by flooding Modbus registers.

One of the main challenges was adapting to unfamiliar Python libraries. While I understood Python well, integrating various simulation tools and cybersecurity implementations required additional learning. Debugging system interactions and ensuring smooth communication between components was an ongoing process, especially when troubleshooting MQTT-based state updates.

Additionally, balancing usability and security within the project required careful thought. Attacks like DoS were effective in disrupting the system, but they also exposed vulnerabilities in real-time communication protocols. Implementing mitigation techniques helped solidify my understanding of defensive cybersecurity measures, reinforcing key principles like rate limiting and structured logging.

This project broadened my perspective on Python's versatility beyond its common use in web application backends. Working with Flask was an intuitive experience, given my background in Django, and seeing Python applied in a cybersecurity context highlighted how adaptable it is for secure system design. The project also reinforced the importance of modular design, having clear boundaries between process control, security layers, and UI interactions helped streamline debugging and collaboration.

Moving forward, I'd like to deepen my knowledge of cybersecurity attack patterns and defensive frameworks. Learning more about real-world ICS security vulnerabilities would help me refine future implementations, bridging the gap between simulated threats and practical mitigation techniques.

SecureSim was an eye-opening experience that blended simulation, security, and structured problem-solving. This project reinforced the importance of balancing functionality with security, and I'll carry those insights into future work. Seeing Python applied in the realm of industrial cybersecurity has expanded my perspective, and I look forward to continuing to explore security-focused applications in future projects. This was a fun project!