

K-9 Packet Sniffer



Engineering
& Computing

Fall 2024
Senior Design Project

Student: Tiago Muller
Instructor: Seyedmasoud Sadjadi

Florida International University

Pentester / Backend Security Developer

Problem:

Security professionals and network administrators depend on tools like Wireshark for analyzing traffic, yet these tools lack critical user session management. This creates the following challenges:

- Data Security Risks:** Improper file management can result in sensitive information being mishandled, exposing the organization to potential breaches.
- Limited User Accountability:** Without authentication systems, it's impossible to associate sessions with specific users, undermining access control.
- Inefficient File Management:** The reliance on manual saving and organization of capture files increases the likelihood of errors or loss.

Current System:

Tools in Use:

- Popular tools like Wireshark empower users to capture and inspect network traffic in real-time, offering robust packet analysis capabilities.

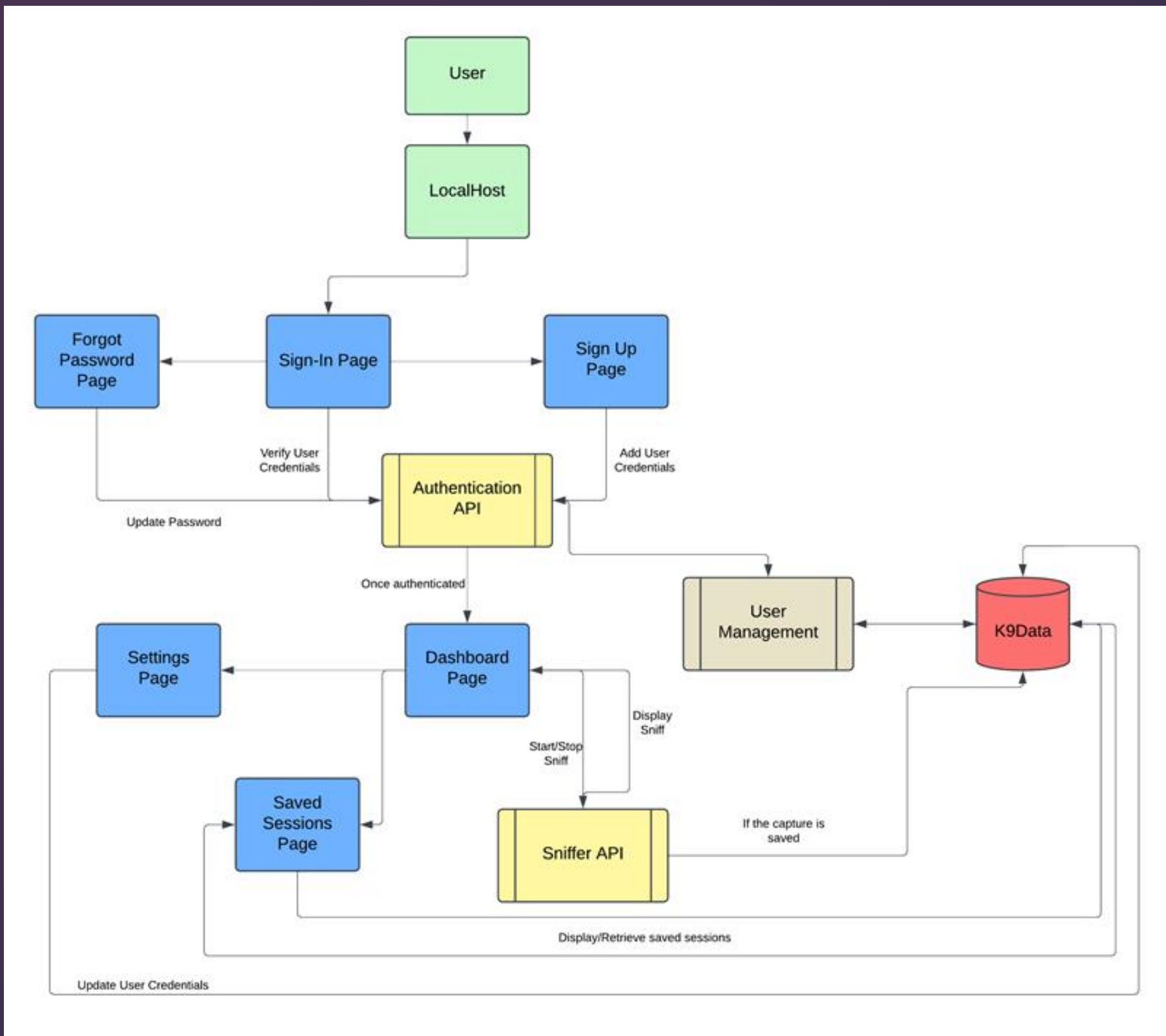
Limitations in Practice:

- Disorganized Workflow:** Capture files must be managed manually, introducing inefficiencies and increasing the likelihood of data mismanagement.
- No Role-Based Security:** Open access to session files means anyone with system access can modify or view them, creating serious security and compliance concerns.
- Scaling Problems:** As the number of capture files grows, managing them becomes impractical, particularly in collaborative or enterprise environments.
- Barriers to Entry:** The interface is not beginner-friendly, with advanced features often hidden behind complex and unintuitive workflows.

Requirements:

- JavaScript
- CSS
- React
- Python
- Scapy
- SQL

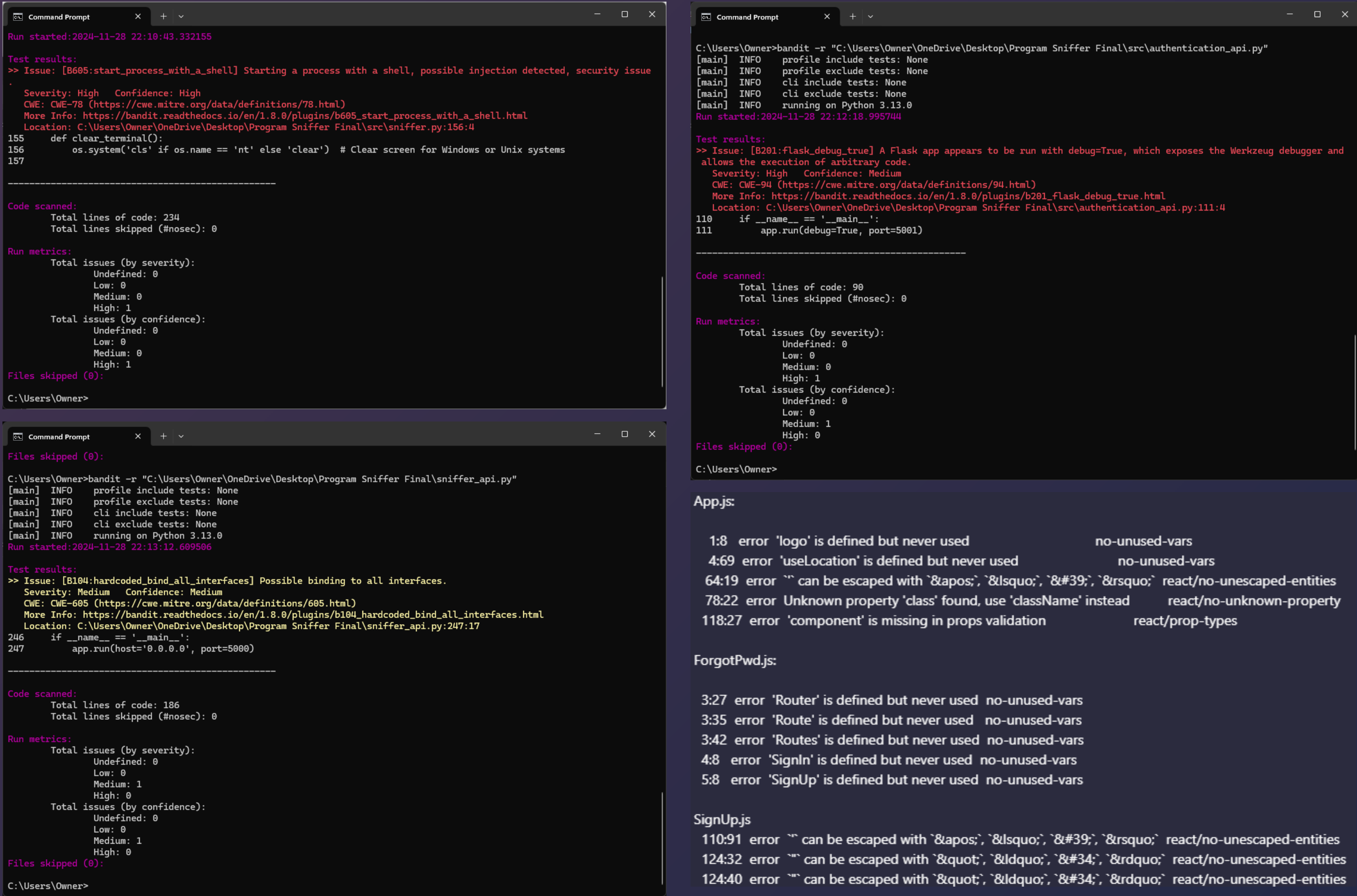
System Design:



Tools Used:



Pentesting Results:



Pentesting Process:

To perform comprehensive testing, we began by conducting a threat and vulnerability analysis using automated scanners to identify known vulnerabilities in dependencies (e.g., Node.js modules) and manually review the source code for potential security issues, focusing on login, authentication, and session management.

Built a risk matrix to prioritize identified threats and tested contingency and mitigation processes, such as secure recovery in the Forgot Password functionality, and validate plans for addressing critical risks.

Ensured data protection by identifying sensitive data, reviewing encryption practices, and verifying access controls. As well as, simulated system failures to evaluate business continuity (BC) and disaster recovery (DR) plans, ensuring documentation and recovery processes are effective.

Risk Matrix:

Risk	Likelihood	Impact	Severity (Priority)
CWE-78: OS Command Injection	High	High	Critical
CWE-94: Code Injection	High	High	Critical
CWE-605: Multiple Port Binds	Medium	Medium	Moderate
Unused Variables (React)	Low	Low	Minor
Unknown Property 'class' (React)	Medium	Medium	Moderate
Unescaped Entities (React)	High	Medium	High
Missing Props Validation (React)	Medium	Medium	Moderate

Acknowledgement

The material presented in this poster is based upon the work supported by Tiago Muller. I am thankful for the help that I received from my group members, Edward Medina, Carlos Imery, Maritza Medina and Sharek Randon