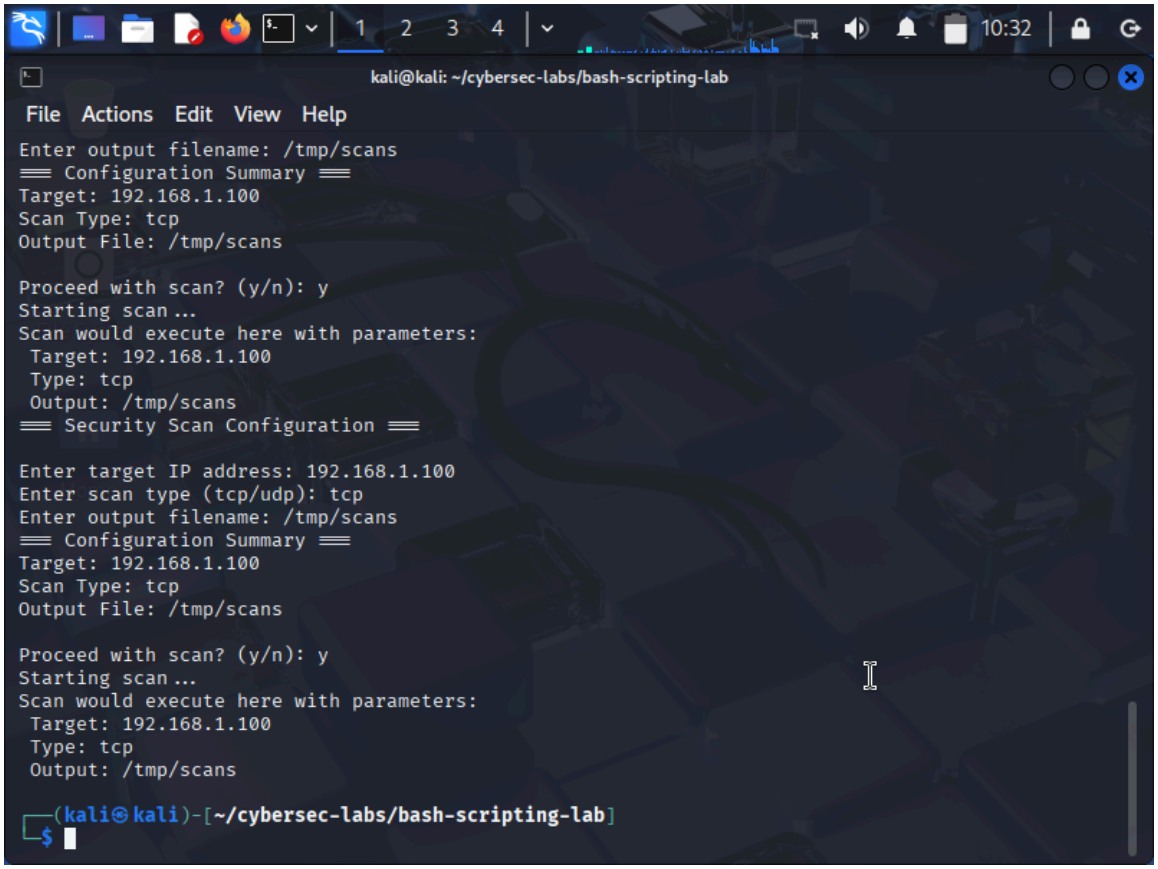
Lab 6: Linux Basics Part 6 - Bash Scripting
Matthew Cox - COMP 325

Introduction:

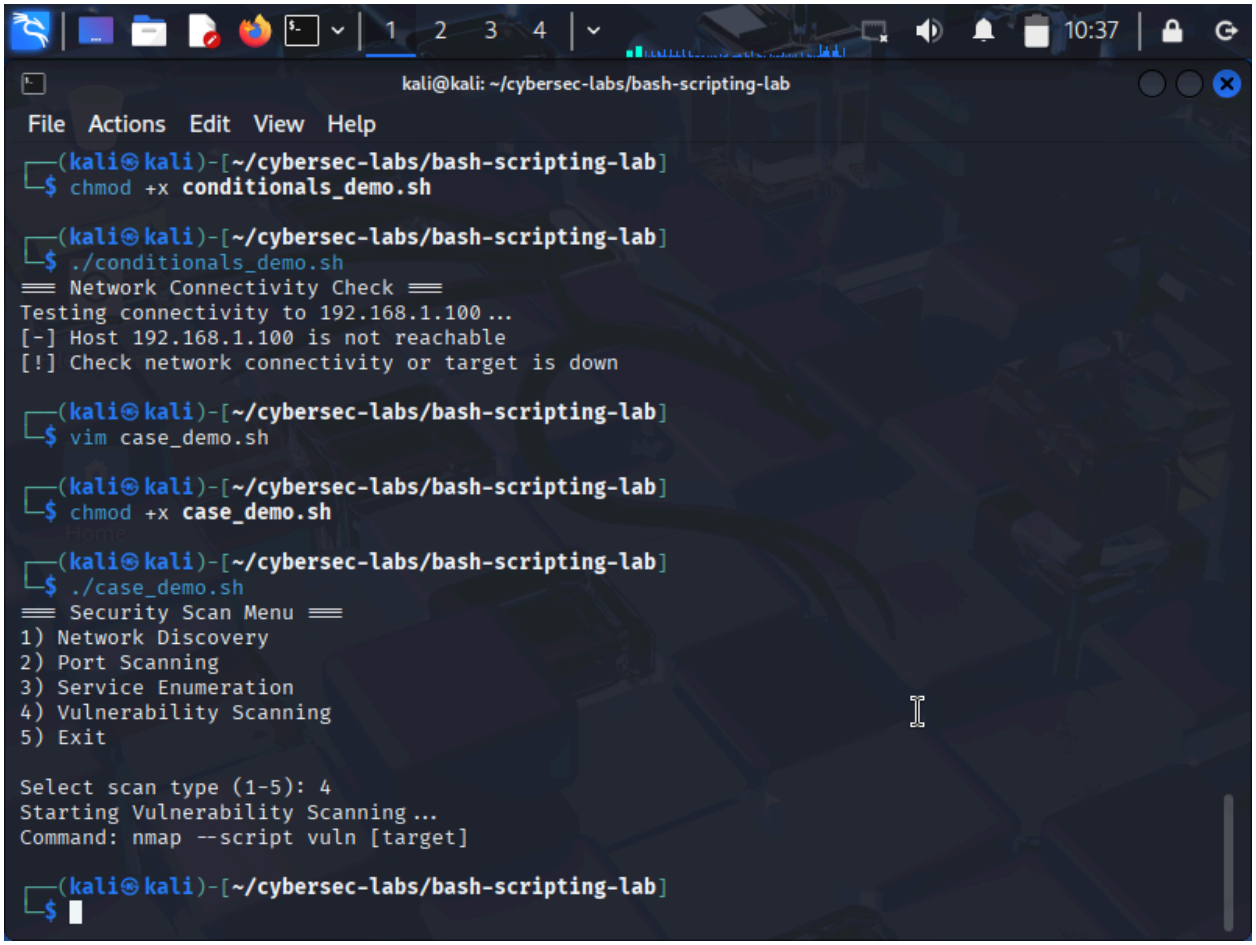
Bash scripting is an indispensable skill for cybersecurity professionals, serving as the foundation for automating repetitive tasks, enabling the rapid deployment of security tools, and facilitating the creation of custom utilities. Security analysts rely on scripts to automate log analysis, monitor system changes, and efficiently process large datasets, while penetration testers use them for reconnaissance automation, exploit development, and post-exploitation activities.

Body:

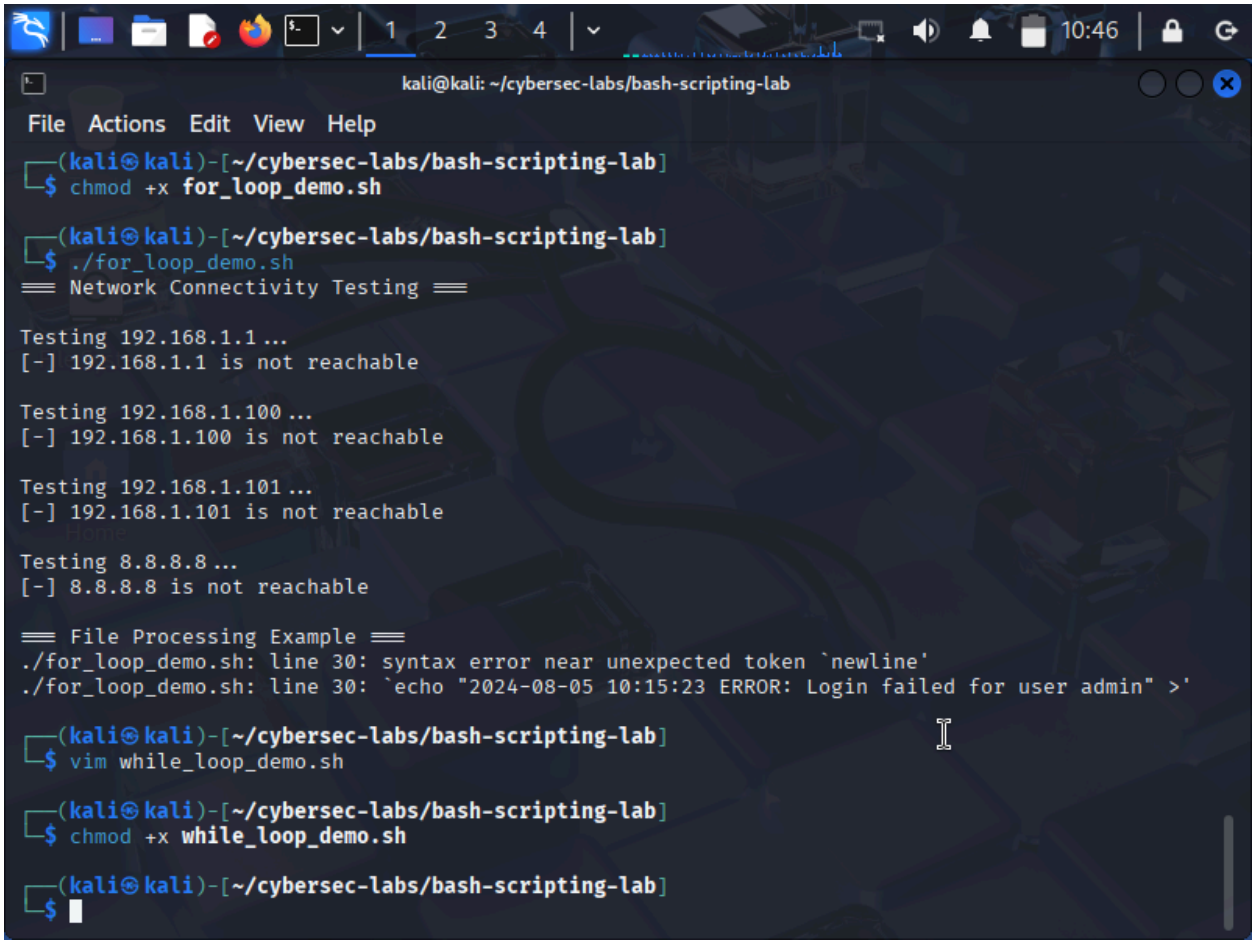


Established foundational scripting knowledge by creating basic security assessment scripts using both nano and vim editors. Implemented variable handling for configuration management, including target IP addresses, scan parameters, and system information collection. Developed interactive scripts with user input validation for security scan configuration.

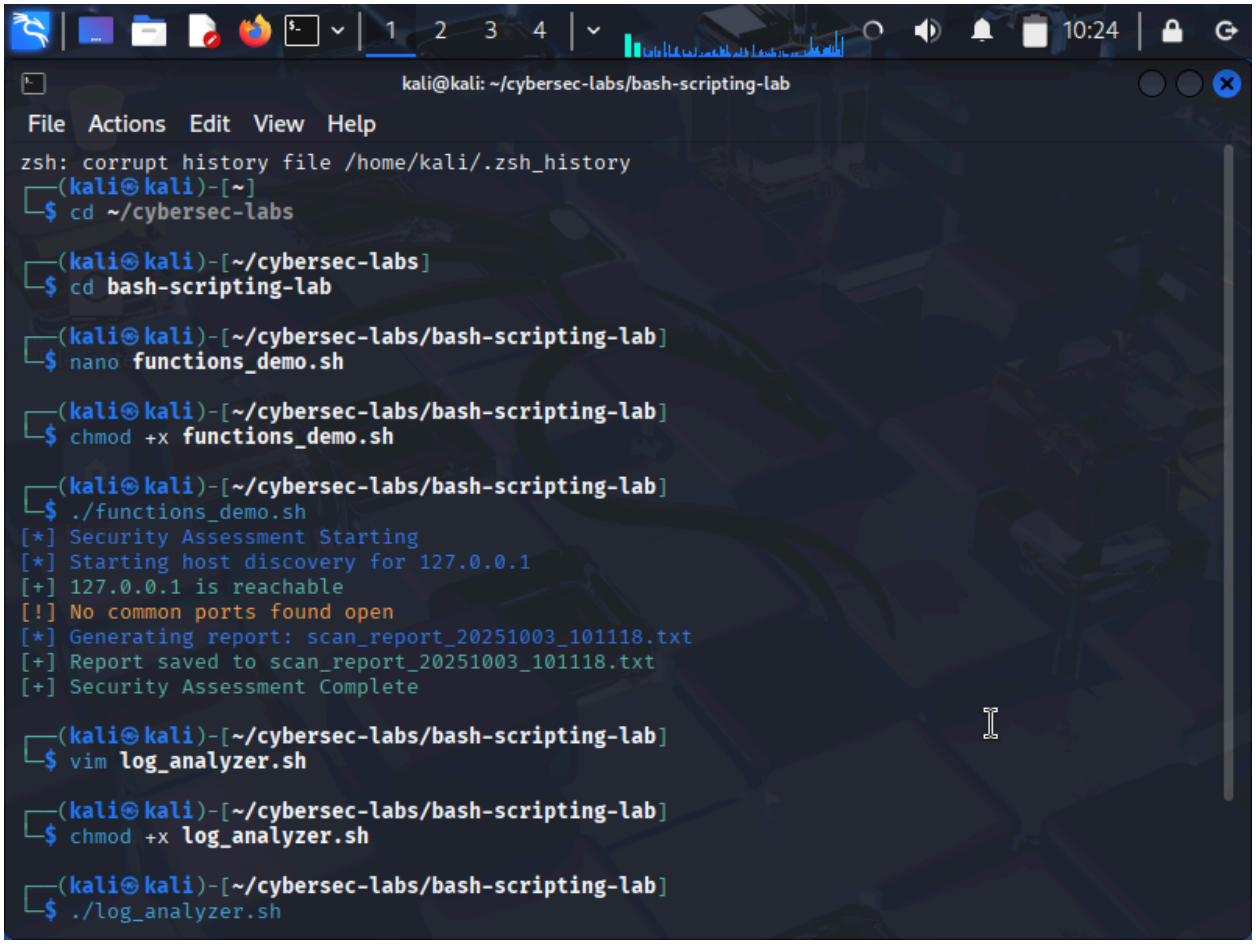
- Shebang notation (#!/bin/bash)
- Command substitution for dynamic data
- Interactive user prompts with validation
- Script structure and documentation standards



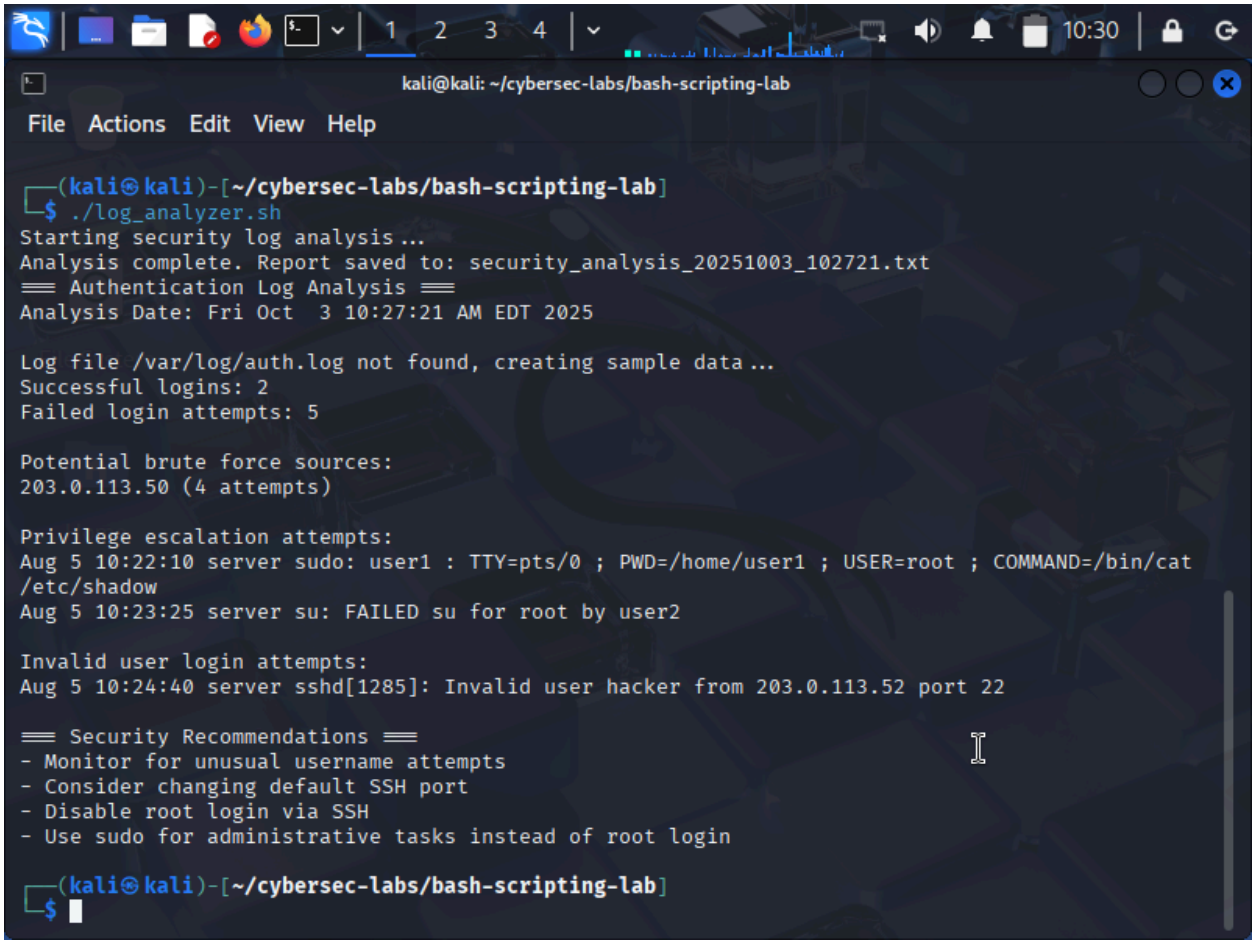
Implemented if-then-else statements and case structures for network connectivity testing and service enumeration. Created scripts that intelligently respond to system states, such as checking host reachability before port scanning and providing contextual recommendations based on discovered services. Technical Challenge: Managing nested conditionals for multi-stage security checks while maintaining code readability.



Developed for loops to iterate through target lists and port ranges, enabling efficient bulk scanning operations. Implemented while loops for continuous security monitoring with time-based constraints. Created log file processing workflows that automatically analyze multiple files for security indicators. Key Achievement: Successfully automated network connectivity testing across multiple targets with parallel port scanning capabilities.



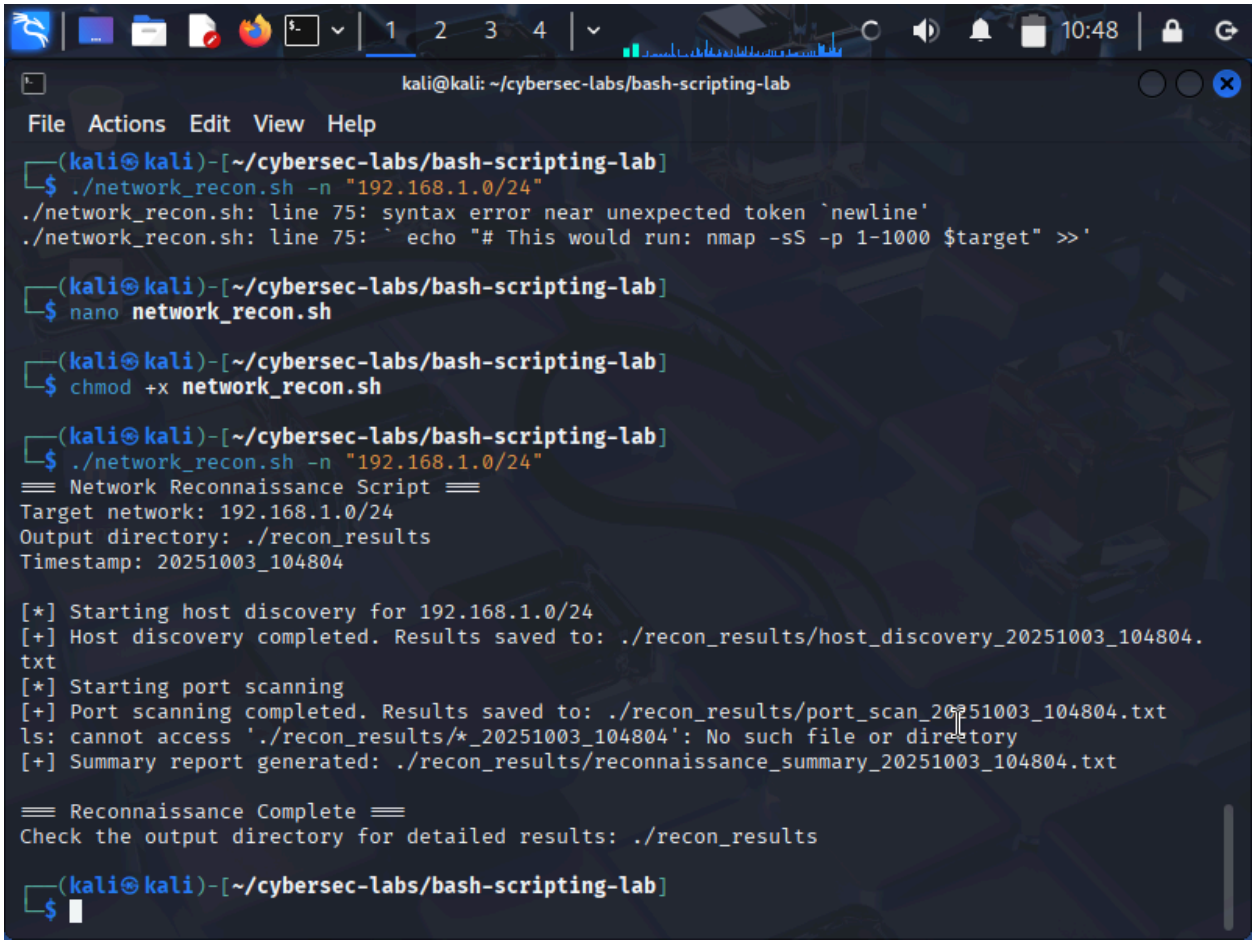
Designed reusable functions for common security tasks, including colored output formatting, port checking, host discovery, and report generation. This modular approach significantly improved code maintainability and enabled rapid development of complex security tools. Implementation Highlight: Created a comprehensive host discovery function that automatically scans common ports and generates formatted reports.



Developed an automated security log analyzer capable of processing authentication logs to identify:

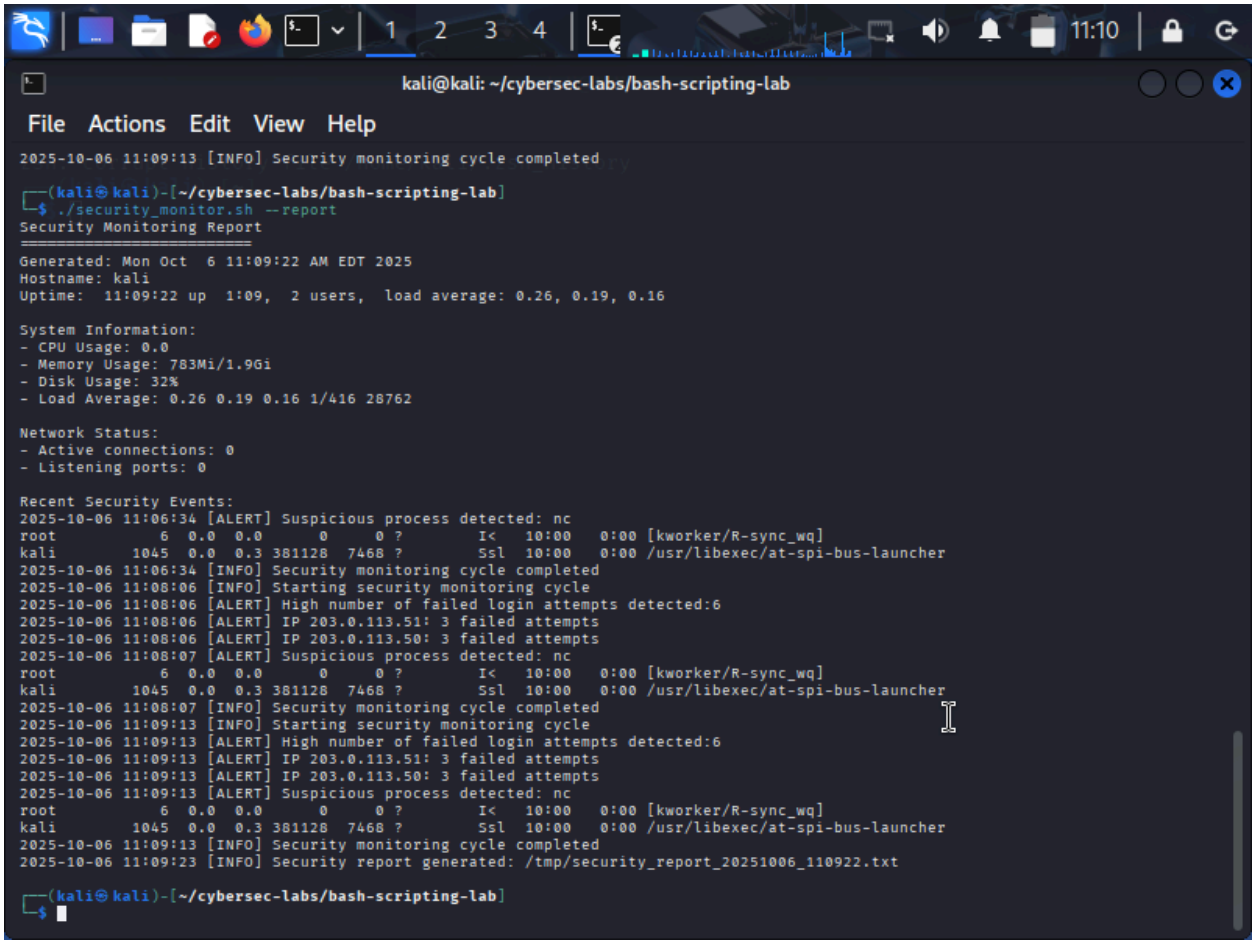
- Failed login attempts and brute force patterns
- Privilege escalation attempts
- Invalid user access attempts
- Security recommendations based on findings

  Technical Difficulty: This exercise presented significant challenges with indentation and script structure, requiring careful attention to syntax for proper execution.



Created a command-line tool for automated network reconnaissance with argument parsing, modular scanning functions, and comprehensive reporting. The script accepts network ranges as input and generates timestamped reports of discovered hosts and open ports.

- Command-line argument parsing with `getopts`
- Output directory management
- Multi-stage scanning workflows
- Summary report generation



Developed the most complex script of the lab—a full-featured security monitoring system with multiple detection capabilities:

- SUID file baseline comparison
- Failed login attempt tracking
- System resource monitoring (CPU, memory, disk)
- Suspicious process detection
- Network connection analysis
- Automated security report generation

Technical Challenges: This exercise combined all previous concepts into a single comprehensive tool. While indentation issues were minimal, several syntax errors and typos required careful debugging. The script supports multiple operational modes (daemon, check, report) for flexible deployment