# Semester Project Final Report - Operating Systems (Spring 2025)

## Course & Submission Information

**Course Title:** Operating Systems

**Semester:** Spring 2025

**Instructor Name:** Asst. Prof. Engr. Abdul Rahman

**Project Title:**

**Real-time Network Intrusion Detection And Monitoring System (NIDMS) Using Raw Sockets in C**

**1. Introduction (Aim or Motivation):**

The surge in cyberattacks, ranging from SYN flood attacks to brute-force login attempts, has necessitated the development of real-time, low-level monitoring tools that can detect anomalies in network traffic. The Network Intrusion Detection and Monitoring System (NIDMS) is designed to analyze network traffic, detect malicious activities, and log potential threats in real time. The system employs multiprocessing, multithreading, synchronization primitives (mutex, semaphores), inter-process communication (IPC) mechanisms (pipes, shared memory), and process scheduling techniques to ensure high performance and responsiveness. This project aims to develop a lightweight, efficient Network Intrusion Detection System (NIDS) implemented in C that directly interfaces with network interfaces using raw sockets. The goal is to detect and alert on suspicious patterns such as SYN flood attacks, port scanning attempts, and brute-force login activities.

**Project Objectives**

* Capture real-time network packets.
* Perform packet inspection for malicious activity (e.g., port scanning, DoS attacks, unauthorized access attempts).
* Use multiprocessing and multithreading to efficiently handle network traffic.
* Implement synchronization techniques (mutexes, semaphores) for safe data handling.
* Implement Producer-Consumer model for processing network packets.
* Address Reader/Writer problem to optimize logging and detection.
* Implement virtual memory management techniques for handling large amounts of data.
* Prevent deadlocks through proper resource allocation strategies.

**2. Background (Research & Project Selection):**

Most modern NIDS systems rely on high-level libraries or pre-built rules. However, in systems with resource constraints or in educational contexts, it's useful to understand how to build such systems from scratch. This project was chosen to explore:

* How packet sniffing works at the raw socket level.
* How multi-threading and inter-process communication can be used for concurrent processing and logging.
* Real-time traffic analysis for anomaly detection.

The design closely reflects academic research and industry practice in terms of detection thresholds and multi-threaded designs.

**3. Project Specification:**

The project involves building a C-based daemon-like application capable of:

* Sniffing network traffic using raw sockets.
* Identifying malicious traffic patterns:
  + SYN flood attacks
  + Port scanning behavior
  + Brute-force login attempts
* Logging alerts asynchronously using inter-process communication.
* Reading logs concurrently without file access conflicts.

**4. Problem Analysis:**

| **Threat** | **Behavior** | **Detection Logic** |
| --- | --- | --- |
| SYN Flood | Multiple SYN packets from same IP in short duration | Track SYN counts in a time window |
| Port Scan | Multiple connection attempts to different ports from same IP | Track distinct ports hit in short duration |
| Brute Force | Repeated connections to login ports (SSH, FTP, Telnet) | Monitor repeated access attempts excluding SYN-only |

Challenges include:

* High-speed packet capturing without packet loss.
* Timely analysis and classification of packets.
* Safe logging in a concurrent environment.

**5. Solution Design (Project Detail, Functionality and Features):**

**Architecture Overview:**

* **Raw Socket Packet Capture:**  
  Reads Ethernet frames using AF\_PACKET and filters IP packets.
* **Multithreaded Producer-Consumer Model:**
  + **Producer Thread:** Captures packets and inserts into a circular buffer.
  + **Consumer Thread:** Analyzes packets from the buffer.
* **Detection Modules:**
  + **SYN Flood Detection:** Tracks SYN packet rate per IP.
  + **Port Scan Detection:** Tracks distinct destination ports per IP.
  + **Brute-Force Detection:** Detects repeated access to login ports with non-SYN-only flags.
* **Logger (Child Process):**  
  Separate process writes alerts to disk via a pipe to avoid blocking the detection threads.
* **Log Reader Thread:**  
  Displays alerts from the log file every few seconds (concurrently, using reader-writer locks).

**Features:**

* Real-time alerts
* Uses raw sockets (no third-party libraries)
* Multi-threaded with synchronization primitives
* IPC via pipes
* Safe concurrent file logging

**6. Implementation & Testing:**

**Tools Used:**

* **Language:** C
* **Libraries:** POSIX Threads, netinet, arpa/inet, semaphore, socket
* **OS:** Linux (tested on Kali Linux)

**Testing Setup:**

* Network interface set to loopback (lo)
* Used nmap, hping3, and hydra for generating:
  + SYN floods
  + Port scans
  + Brute-force attempts

**Results Observed in Terminal:**

* [Timestamp] ALERT: SYN flood from ...
* [Timestamp] ALERT: Port scan detected from ...
* [Timestamp] ALERT: Brute-force attempt detected from ...

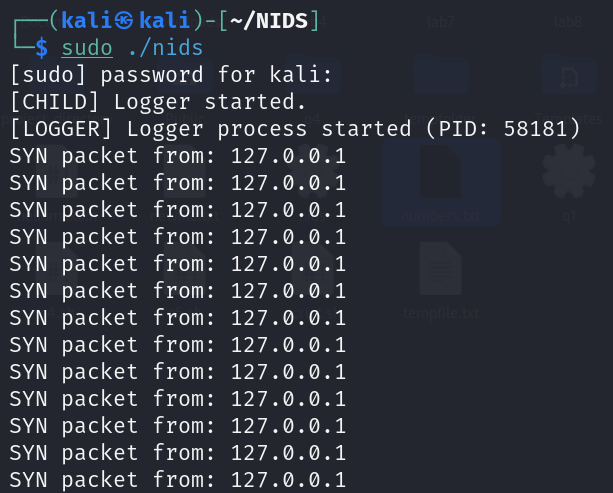
All alerts were also logged to: /home/kali/NIDS/alerts.log

**7. Project Breakdown Structure (Workload distribution with timeline):**

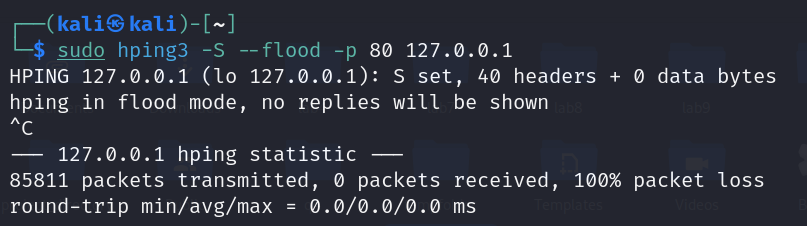
|  |  |  |  |
| --- | --- | --- | --- |
| **Week** | **Phase** | **Tasks & Deliverables** | **Tools/Concepts Used** |
| **Week 8** | Research & Requirements Analysis | - Studied raw socket programming and Linux packet processing - Researched network threats (SYN flood, port scan, brute force) - Defined detection scope | Raw Sockets, TCP/IP model, Intrusion Detection Basics |
| **Week 9** | Design Architecture & Structures | - Designed system architecture: multi-threaded with producer-consumer model - Chose synchronization methods (mutexes, semaphores) - Sketched detection logic | Architecture Diagrams, Threading Design, Data Structures |
| **Week 10** | Implement Packet Capturing and Buffer Logic | - Built raw socket sniffer - Parsed Ethernet, IP, and TCP headers - Implemented circular buffer with mutex+semaphore coordination | AF\_PACKET, pthread, Circular Buffer, sem\_t |
| **Week 11** | Implement Detection Modules | - Added SYN flood detection (rate-based) - Developed port scan detection (distinct ports logic) - Implemented brute-force tracker (login port monitoring) | Time-based detection, Port tracking, TCP analysis |
| **Week 12** | Add Logging & IPC | - Forked child logger process using fork() - Used pipe for alert communication - Implemented Reader-Writer model for concurrent log file access | pipe(), fork(), Reader-Writer Synchronization |
| **Week 13** | Testing and Validation | - Used nmap, hping3, hydra for simulating attacks - Verified correct alerts and logging - Tuned thresholds and time windows | Pen-testing tools, Debugging, Live Packet Analysis |
| **Week 14** | Report Preparation | - Documented project report with architecture, code breakdown, and screenshots - Structured content as per project guidelines | Technical Writing, Report Structuring, Screenshots |

**8. Results (Outputs):**

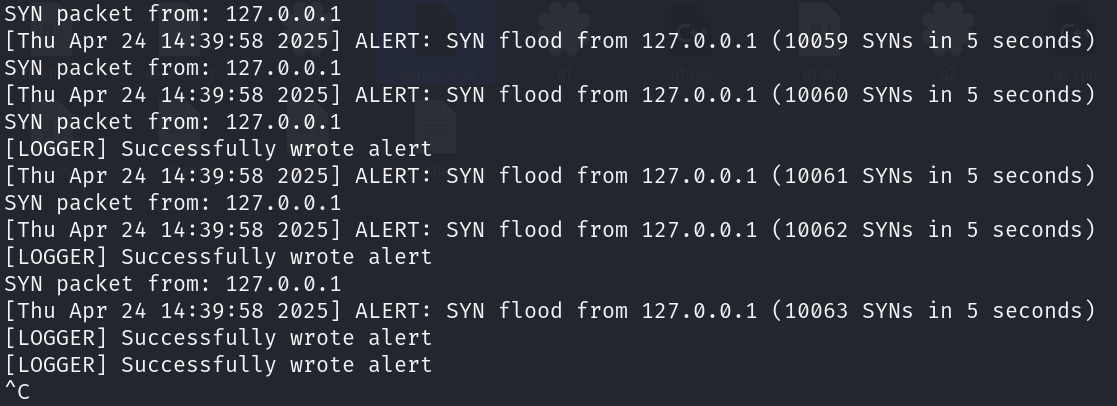
Packet Sniffing…



SYN Flood Attack



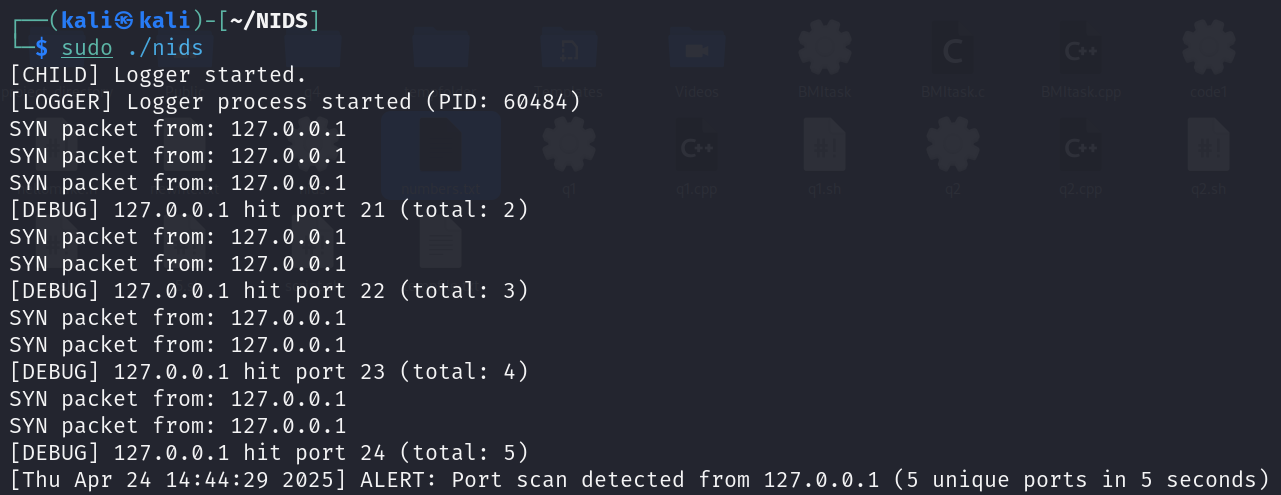
Alerts



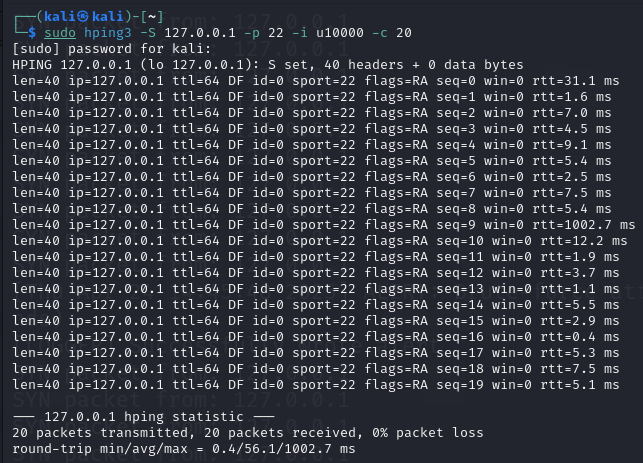
Port Scan Attack



Alerts



Brute Force Attack

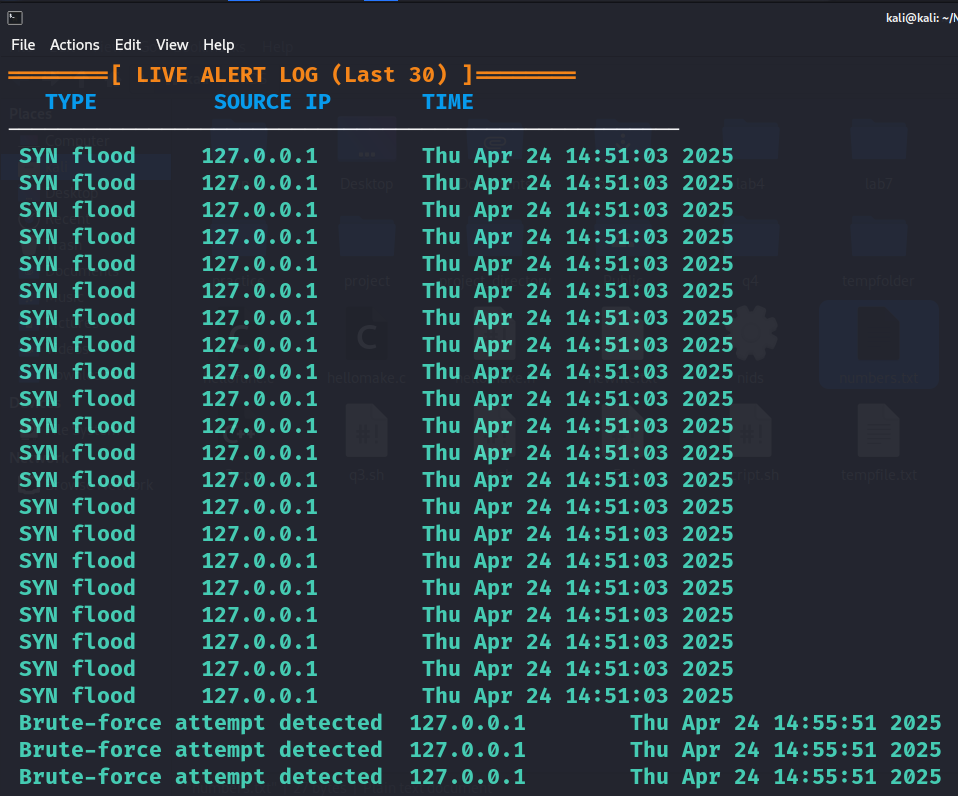


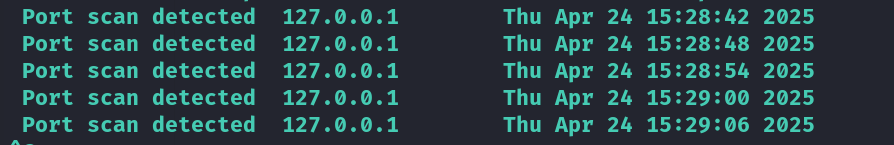
Alerts



Separate File for Alerts(alerts.log)

Output of alerts\_display.c file stores last 30 attacks:





**9. Conclusion (Summary & Discussion):**

This project successfully demonstrates the power and efficiency of a low-level NIDS implemented in C using raw sockets and multithreading. The detection mechanisms are lightweight, real-time, and suitable for academic, experimental, or constrained environments. This project provides with with hands-on experience in network security, OS concepts, and concurrent programming, making it an excellent real-world systems programming project. While simplistic compared to full-fledged tools like Snort or Suricata, it provides excellent insight into core concepts such as:

* Packet parsing
* Pattern recognition
* Concurrent processing
* System-level IPC and file handling

**Future Improvements:**

* Add support for IPv6
* Use PCAP library for portability and better packet parsing
* Integrate with email/SMS alert systems
* Add anomaly-based detection using ML techniques

**Group Information:**

|  |  |
| --- | --- |
| **Name** | **Roll Number** |
| Yash Lal (Group Leader) | 23K-0519 |
| Asad Ali | 23P-0699 |
| Muhammad Humayun Dar | 23K-0795 |