NAAC Accredited & UGC 12 (B) Status Holder

Abstract and Objectives of

Intrusion Detection System

Submitted by:

Name(Student 1):	**Abhishek Dutta**
Name(Student 2):	**Chinmoy Das**
Enrollment (Student 1):	**ADTU/2022-25/BCASP/040**
Enrollment ID(Student 2)	**ADTU/2022-25/BCASP/014**
Semester:	**6th Semester**
Section:	**A**

Submitted to:

Faculty In-charge	**Dr. Mala Dutta**

Abstract

This project proposes a hybrid Intrusion Detection System (IDS) that fuses the robust packet inspection capabilities of **Snort** with the simplicity and automation potential of **Python scripting**, offering a more accessible and efficient solution for network threat detection. Recognizing the complexity involved in manually configuring and monitoring Snort—especially in Windows environments—the project shifts to a Linux-based architecture (via **WSL with Kali Linux**) for greater control and performance.

Snort is compiled from source using **CMake**, incorporating a suite of open-source dependencies such as **libdaq**, **libdnet**, **PCRE2**, **OpenSSL**, and **LuaJIT**. While Snort handles low-level packet capture and rule-based inspection, Python is employed to manage high-level automation tasks—such as initiating scan detection, analyzing logs in real-time, and displaying alerts through a streamlined start/stop interface. This modular architecture ensures not only improved user experience but also facilitates scalability, maintenance, and integration into broader cybersecurity frameworks.

Objectives

I. To develop a high-performance, real-time IDS using Snort as the core engine, combined with a Python-based interface for automation and usability.

II. To compile Snort from source using C++17 and configure it with essential dependencies such as:

a) **libdaq** – Packet I/O handling,

b) **libdnet** – Low-level network tasks,

c) **LuaJIT** – Scripting support for flexible configuration,

d) **PCRE2** – Advanced pattern matching for detecting intrusion patterns,

e) **OpenSSL** – For secure communication and hash-based verification,

f) **zlib** – To support compressed payloads,

g) **hwloc** – For CPU and memory affinity management.

III. To create a Python-based GUI that enables:

a) One-click start/stop monitoring,

b) Real-time log visualization using tools like tail and subprocess,

c) Custom rule creation and integration.

IV. To implement modular, scalable architecture where Snort handles packet-level detection and Python handles alert processing, UI, and configuration logic.

V. To enhance detection of scanning tools like **Nmap**, suspicious TCP flag patterns, and other known vulnerabilities through a customized Snort ruleset.

VI. To reduce operational complexity and false positives through rule tuning and automated log filtering.