# Proof Of Concept: NIDS

**Members:** Rishikesh Tripathi (249), Tanaya Suryawanshi (305)

## IDS Overview:

- **What are IDS?** → A cybersecurity system that monitors network or host activity to detect malicious actions and policy violations.
- **Why is IDS important?** → It helps detect attacks early, reduce damage, and provide alerts for quick response.

## Subtopics in IDS:

- **Types of IDS**:

Network-based (NIDS): Monitors network traffic in real-time to detect malicious patterns.

Host-based (HIDS): Monitors a specific device/server's logs, files, and processes for suspicious activity.

- **Detection Techniques** :

Signature-based: Uses predefined attack signatures to detect known threats.

Anomaly-based: Identifies deviations from normal behaviour to catch unknown threats.

Hybrid detection: Combines signature and anomaly-based techniques for better accuracy.

- **Deployment Models**:

Inline IDS: Placed directly in the data path to analyse traffic before forwarding.

Passive IDS: Observes network traffic through a tap or span port without interfering.

- **Key Components**:

Sensors: Components that collect data packets or logs from the environment.

Analysers: Systems that process collected data and identify suspicious activity.

Management Console: Centralized interface for configuring IDS and reviewing alerts.

Alert System: Mechanism that notifies administrators when intrusions are detected.

• **IDS vs IPS**:

IDS detects & alerts: IDS only detects and alerts.

IPS detects & prevents: IPS detects and actively blocks attacks.

• **Integration with SIEM**:

IDS feeds log to SIEM for centralized analysis.

• **Open-Source IDS Tools**:

Snort, Suricata, OSSEC, Bro/Zeek.

• **Future Trends**:

AI-driven IDS: Advanced IDS using machine learning to detect zero-day and complex threats.

Cloud-native IDS: IDS designed specifically to secure workloads in cloud environments.

Behaviour-based Analytics: IDS that detects threats by analysing user and system behaviour patterns.

## Who Needs IDS?

- **Enterprises & Corporates** → To protect sensitive business data from hackers.
- **Banks & Financial Institutions** → To secure online transactions and prevent fraud.
- **Government & Defence** → To safeguard national security systems.
- **Cloud Service Providers** → To ensure client data safety in shared environments.
- **Educational Institutions** → To protect research data and student records.
- **Individuals (Tech-Savvy Users**) → For added host-level protection at personal level.

## Who Should Learn IDS?

- **Cybersecurity Students & Researchers** → To understand intrusion detection **concepts.**
- **Network Administrators** → To monitor and secure organizational networks.
- **Ethical Hackers & Pen-Testers** → To simulate attacks and test IDS effectiveness.
- **Incident Response Teams** → To analyse alerts and handle breaches.
- **Security Engineers & SOC Analysts** → To operate IDS tools in real time.

## Where are IDS Applied & Who Benefits?

- **Data Centres** → For detecting attacks on servers and virtual machines.
- **Enterprise Networks** → For preventing internal/external breaches.
- **Healthcare Systems** → To protect patient data from ransomware.
- **E-commerce Platforms** → To prevent card fraud and account takeover.
- **Cloud & Hybrid Environments** → To secure distributed workloads.
- **Benefit** → Organizations, customers, and governments gain trust, compliance, and security.

# Main Tool:

## Libraries Used

- **tkinter**: Used to build the graphical user interface (GUI), including the main window, frames, buttons, and display tables.
- **scapy**: The core packet manipulation library used to capture, dissect, and analyse network packets from a network interface.
- **threading**: Allows the packet sniffing process to run in a separate background thread, preventing the GUI from freezing during capture.
- **queue**: Provides a thread-safe way to pass captured packets from the background sniffing thread to the main GUI thread for analysis and display.
- **time**: Used for generating timestamps for packets and for implementing the time window logic in the port scan detection.
- **collections.defaultdict**: Used to efficiently track the unique ports contacted by each source IP address for port scan detection.

### Core Components & Logic

### Intrusion Detection Rules

- **SUSPICIOUS_KEYWORDS**: A list of byte strings (e.g., b'SELECT', b'<script>') that are checked against the raw payload of packets. Their presence can indicate potential SQL Injection or Cross-Site Scripting (XSS) attacks.
- **BLACKLISTED_IPS**: A set of known malicious IP addresses. Any traffic from these Ips is immediately flagged.
- **Port Scanning Detection**:
    - PORT_SCAN_TRACKER: A dictionary that stores the set of destination ports a source IP has tried to connect to.
    - PORT_SCAN_TIME_WINDOW: The time frame (in seconds) within which connections are monitored.
    - PORT_SCAN_PORT_COUNT: The threshold for the number of unique ports contacted within the time window to trigger an alert.

### Function Breakdown

- **NIDS.__init__(self, root)**: Initializes the main application class, sets up the GUI window, themes, and variables.
- **NIDS._define_themes(self)**: Defines the color schemes for the "dark" and "light" UI themes.
- **NIDS._apply_theme(self)**: Applies the currently selected theme's colors to all GUI widgets.
- **NIDS.toggle_theme(self)**: Switches between the light and dark themes.
- **NIDS._create_widgets(self)**: Creates and arranges all the GUI elements like buttons, labels, and the treeview tables for displaying packets.
- **NIDS.start_sniffing(self)**: Initiates packet sniffing by starting a new background thread and updating the GUI status.
- **NIDS.stop_sniffing(self)**: Sets a flag to stop the background sniffing thread and updates the GUI status.
- **NIDS.packet_sniffer(self)**: The function that runs in the background thread. It uses scapy.sniff() to capture packets.
- **NIDS.process_packet(self, packet)**: A callback function for scapy.sniff(); it takes each captured packet and puts it into the packet_queue.
- **NIDS.analyze_packet(self, packet)**: The core detection engine. It checks each packet against the defined rules (blacklisted Ips, suspicious keywords, port scanning) and returns a threat score and a summary.
- **NIDS.update_gui(self)**: Runs on a timer in the main thread. It pulls packets from the queue, sends them to analyze_packet(), and inserts the results into the "Live Packet Stream" and "Threat Detections" tables.

- **NIDS.show_packet_details(self, event)**: Triggered when a user clicks on a packet in either table. It displays the full, detailed content of the selected packet and provides mitigation advice for detected threats.

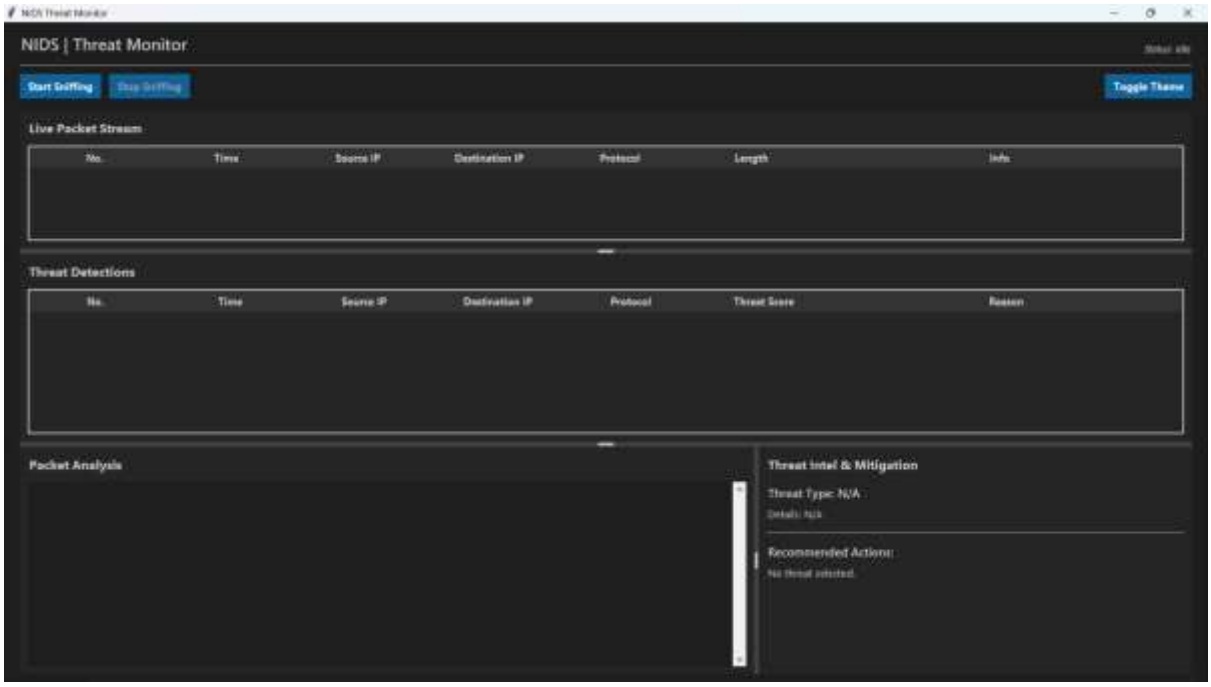
## Working:

First the tool is run via Administrative permissions in order to capture packets and analyse them.

We can do that in two manners, by either starting the program in an administrator privileged PowerShell or running the script and then giving it administrative permission.

**Step 1:**

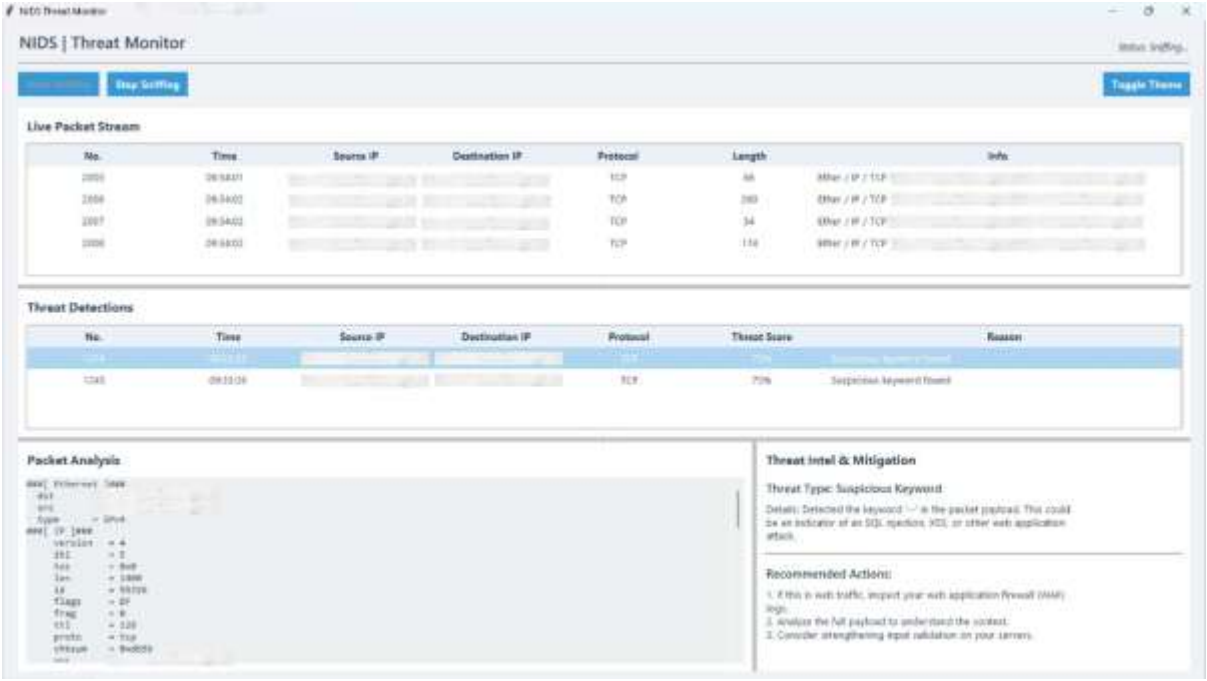
When our tool starts, we can see a “Start Sniffing” button, clicking it starts the packet capturing and analysis




Here we can clearly see the sections, the topmost is for the ongoing packet traffic that happens, after that in the middle there is a threat detection section where the potential threats will be marked out, and below in the left side there is a packet analysis section which shows the details about the packet, and just the immediate right section is for threat intel, meaning it will try to flag out what the threat is, and how should we deal with it.

**Step 2:**

After starting the Packet sniffing process, it starts to capture and analyse all those.




Here we can clearly see how the packets are being captured and analysed, and if a potential threat is found, we get the details and measures as to how we can keep our systems safe.

And hence this tool is a working NIDS. Which would help us keep an organization safe.

## Advantages & Disadvantages:

- **Advantages**: Early detection of intrusions, supports forensic analysis, complements firewalls, increases compliance, detects insider threats.
- **Disadvantages:** High false positives, requires skilled staff, doesn't block attacks (only detects), performance overhead, costly for small businesses.

## Need for Advances in IDS:

- **AI & ML Integration** → To reduce false positives and detect unknown threats.
- **Cloud-native IDS** → To protect workloads in dynamic cloud environments.
- **Encrypted Traffic Analysis** → To detect threats inside TLS/SSL traffic.
- **IoT & OT Security** → To protect smart devices and industrial systems.
- **Automated Response (IDS + SOAR)** → To not just alert but auto-mitigate attacks.