International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 05 | May - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

Real Time Network-Based Intrusion Detection System

Authors: Harshit Gupta, Shahid Ali, Sumeet Negi, Himanchal Kumar Gond

Department of Computer Science & Engineering

Babu Banarasi Das Northern India institute of Technology, Lucknow

Affiliated to the

DR. APJ ABDUL KALAM TECHNICAL UNIVERSITY LUCKNOW

1. ABSTRACT

This research explores a real-time, custom-built Network Intrusion Detection System (NIDS) capable of live packet capture, visualization, and threat detection based on known malicious patterns such as command injection, DNS tunnelling, and reverse shells. It includes practical simulation tools for demonstrable cybersecurity education. This system integrates live packet sniffing with an intelligent pattern recognition engine and uses behaviour-based indicators to detect potential intrusions in Wi-Fi-based environments. Through testing with real and simulated traffic, the tool provides a reliable, demonstrable NIDS model for academic and small enterprise use.
The system is lightweight, requires minimal setup, and operates in cross-platform environments including Windows and Linux. It supports modular extensions for future research, including integration with machine learning models and remote alerting systems.

2. INTRODUCTION

As cyber threats continue to grow in sophistication, the need for effective intrusion detection systems (IDS) has never been more pressing. Traditional host-based detection methods are limited, as they often fail to detect attacks that happen across networks in real-time. With cybercriminals employing increasingly advanced techniques, it's vital to have systems in place that can monitor, analyse, and respond to these threats instantly. This project aims to address this gap by proposing a real-time Network Intrusion Detection System (NIDS). Drawing inspiration from widely-used network analysis tools like Wireshark, this system will enhance the detection and monitoring process. However, unlike conventional tools, the proposed NIDS will feature automated threat detection and alerting capabilities. This will allow for immediate identification of suspicious activities and reduce the response time. Furthermore, the system will include simulation features that allow security professionals to test and assess its effectiveness in various attack scenarios. Through these innovations, this project seeks to provide a more efficient, adaptive, and scalable solution for modern cybersecurity challenges.

- **Why is this study important?**

This study is important because it provides a practical and cost-effective solution for detecting suspicious network activities using a lightweight, customizable Intrusion Detection System (IDS). It helps individuals and small organizations understand and respond to cyber threats in real time. The tool mimics professional systems like Wireshark, making it useful for education and training. By simulating and identifying real-world attacks, it enhances cybersecurity awareness. Ultimately, it bridges the gap between theoretical learning and real-world network defence.

- **What has been done before?**

Many intrusion detection systems (IDS) have been developed over the years, with a primary focus on either network-based or host-based detection. Tools like Wireshark and Snort are well-known for network traffic analysis and intrusion

detection, providing insight into potential threats. However, these tools often rely on manual intervention or require significant configuration to detect emerging threats. Other solutions, such as Suricata, offer real-time network monitoring and anomaly detection but still face limitations in automated threat identification and response. Some modern IDS frameworks have incorporated machine learning and AI for threat detection, but they are not always tailored to specific real-time use cases. Existing systems also lack simulation features to assess the effectiveness of detected threats in varying attack scenarios. Despite advances, many of these systems struggle with false positives and lack the ability to simulate real-world attacks for testing. This project seeks to improve upon these limitations by providing an automated, real-time NIDS with enhanced alerting and simulation capabilities.

- **Objective of this study**

To develop and demonstrate a network-based IDS that can capture, analyse, and detect suspicious activity in real-time, while also simulating such attacks for educational and training purposes. Traditional firewalls and antivirus systems fail to detect sophisticated threats that occur within network layers. This research bridges the gap by developing a real-time, user-friendly tool capable of deep packet inspection and anomaly detection, backed by live testing against simulated malicious activity for academic realism and reproducibility.

---

3. MATERIALS AND METHODS

Materials

The development and testing of the NIDS involved two core phases: **live packet capturing** and **real-time suspicious activity detection.**

Tools & Libraries Used

The development of this Network Intrusion Detection System (NIDS) was made possible through the integration of several Python libraries and modules, each serving a specific role in real-time traffic analysis and alerting:

- **Scapy:** A powerful Python-based packet manipulation library used to sniff, decode, and analyse network traffic at multiple protocol layers (Ethernet, IP, TCP/UDP, DNS, etc.). It enables packet filtering, inspection, and modification essential for NIDS.
- **Tkinter:** Python's standard GUI toolkit was employed to build an interactive and visually intuitive user interface that replicates the core visual functionalities of tools like Wireshark**.**
- **Plyer:** A cross-platform notification library used to generate real-time desktop alerts when potentially malicious activity is detected. It ensures immediate user awareness of threats.
- **Threading and Queue Modules:** To support smooth GUI operations while capturing and analysing packets in the background, threading was used to separate packet sniffing from the UI rendering, improving responsiveness and performance.
- **Custom Simulation Scripts:** Python scripts were developed to simulate benign and malicious traffic patterns, including reverse shell signatures, SYN floods, and encoded DNS queries. These simulations provided realistic testing conditions without compromising system security.

System Workflow

The NIDS follows a carefully structured and event-driven workflow, ensuring both usability and comprehensive threat coverage:

1. **Network Interface Selection:** At launch, the system scans and lists available network interfaces, allowing the user to select the appropriate interface for live traffic monitoring (e.g., the laptop's active Wi-Fi connection).

2. **Live Packet Capture:** Using scapy.sniff(), the system initiates packet capture on the selected interface, continuously collecting incoming and outgoing traffic across various protocols.

3. **Pattern-Based Detection Engine:** Each captured packet is inspected for known suspicious behaviours. These include payloads associated with common attack vectors such as cmd.exe, PowerShell, or uncommon port usage indicating potential reverse shell attempts. DNS tunnelling signatures are also evaluated.

4. **Graphical Packet Display:** The GUI updates in real time, displaying captured packets with essential details like source/destination IP, protocol, and payload snippet. Protocols are color-coded for quick identification (e.g., red for TCP, green for UDP, blue for DNS).

5. **Real-Time Desktop Alerts:** If a packet matches predefined malicious patterns, Plyer triggers a native system notification, alerting the user even if the GUI is minimized.

6. **Logging and Archival:** All captured traffic, along with detection results, are optionally saved in pcap format for later review or forensic analysis.

Additional Features

To emulate an enterprise-level IDS, several auxiliary features were included to enhance practicality and maintainability:

- **Packet Archiving Support:** Captured data can be exported in .pcap format compatible with Wireshark or other forensic tools, supporting post-incident investigation.
- **GUI-Based Alert History:** The system maintains a scrollable alert log within the interface, allowing security personnel to view a timeline of suspicious activity.
- **Multithreaded Processing:** By running the capture engine and detection logic in separate threads, the system ensures that GUI operations such as scrolling and filtering remain responsive.
- **User-Centric Design:** Emphasis on simplicity and usability makes this system ideal for educational, training, and lightweight production use in small to medium-sized environments.

Attack Simulation and Realistic Testing

To validate the effectiveness of the NIDS, a safe and controlled attack simulation environment was developed. The test environment included:

- **Reverse Shell Pattern Emulation:** Custom Python scripts generated traffic with encoded payloads typically seen in remote shell attacks.
- **SYN Flood Testing:** A flood of TCP SYN packets was sent to simulate a Denial of Service (DoS) condition and validate volume-based detection.

- **Encoded DNS Queries:** Queries using abnormal subdomain structures and uncommon top-level domains were tested to simulate DNS tunnelling behaviour.

.

4. RESULTS

- **Detection Accuracy:** The system successfully detected simulated command injection attempts and DNS tunnelling activities with consistent precision. The detection engine reliably matched suspicious patterns within packet payloads against predefined attack signatures.

- **Graphical User Interface Feedback:** The GUI responded in real-time to incoming traffic, dynamically displaying packet attributes such as source and destination IP, protocol type, and summarized payload data. Protocols were visually distinguished through colour coding, aiding in intuitive monitoring.

- **Simulated Attack Demonstration**: A custom attack simulation script was executed to inject realistic reverse shell and abnormal DNS payloads into the network stream. The system immediately generated alerts for these patterns, confirming its efficacy in live conditions.

- **Packet Archival and Validation:** All captured traffic, including suspicious packets, was stored in PCAP format. Subsequent analysis with Wireshark verified the accuracy of the captures, ensuring the tool maintains forensic compatibility.

- **Alert Verification:** Alerts raised by the NIDS were manually cross-referenced with pre-logged attack vectors. Each alert was validated as a true positive, demonstrating the precision of the detection logic without false positives in the test set.

- **Usability Assessment**: Informal usability testing with a group of university cybersecurity students revealed high levels of engagement and satisfaction. The real-time visualization and instant feedback made the tool accessible for educational and training purposes.

5. DISCUSSION AND CONCLUSIONS

This research project presents a functional and educationally valuable Network Intrusion Detection System (NIDS) that integrates real-time packet sniffing, suspicious pattern detection, and a user-friendly graphical interface. It serves both as a demonstration tool for academic purposes and a prototype for lightweight enterprise network security applications.

Educational Value and Practical Application

The system is designed to offer an accessible introduction to IDS concepts without requiring enterprise-scale infrastructure. Through its interactive GUI and real-time feedback, users can observe packet flow, detect simulated threats, and understand the foundational mechanics of intrusion detection systems. It is especially well-suited for cybersecurity coursework, lab environments, and hands-on demonstrations.

Limitations

Despite its practical utility, the current system has some constraints:

- **Signature-Based Detection:** Relying on known patterns can lead to false positives or miss zero-day threats.

- **Scalability Issues**: High-throughput networks may experience latency or packet loss without optimized capture engines.
- **Limited Protocol Coverage:** The system currently targets basic TCP, UDP, and ICMP protocols and may not fully support encrypted or tunnelled traffic.
- **Local-Only Logging:** Without backend integration, alert logs are local and not centralized, limiting broader analysis.

Future Work

To enhance the effectiveness and scalability of the system, the following improvements are proposed:

- **Machine Learning Integration:** Implementing anomaly detection models to identify unknown or evolving threats.
- **Database Integration:** Logging suspicious activity into structured databases (e.g., SQLite, PostgreSQL) for long-term analysis.
- **Remote Alerting**: Sending logs to a Security Operations Centre (SOC) via secure channels for centralized threat monitoring.
- **Encrypted Traffic Analysis:** Exploring deep packet inspection or metadata analysis to identify threats in SSL/TLS streams.
- **Auto-PCAP Scheduling**: Adding functionality for timed packet capture exports for scheduled threat audits.

Conclusion

The project demonstrates the viability of combining traditional packet inspection techniques with modern user interface design to create a hybrid IDS. By incorporating live simulation scripts, the system allows real-time threat generation and response, offering a hands-on view of NIDS operation. This approach enhances learning, improves situational

---

6. REFERENCES CITED

- Scapy. (2023). *Scapy Documentation*. Retrieved from https://scapy.readthedocs.io/en/latest/
- Lippmann, R., Fried, D. J., Graf, I., Haines, J. W., & Kendall, K. R. (2000). *Evaluating Intrusion Detection Systems: The 1998 DARPA Off-Line Intrusion Detection Evaluation*. Proceedings of the DARPA Information Survivability Conference and Exposition, 2000, 12–26.
- Paxson, V. (1999). *Bro: A System for Detecting Network Intruders in Real-Time*. Computer Networks, 31(23-24), 2435–2463.
- Sommer, R., & Paxson, V. (2010). *Outside the Closed World: On Using Machine Learning for Network Intrusion Detection*. IEEE Symposium on Security and Privacy, 2010, 305–316.
- Tkinter. (2023). *Python Tkinter GUI Programming*. Python Software Foundation. Retrieved from https://docs.python.org/3/library/tkinter.html
- Plyer. (2023). *Plyer Cross-platform Notifications for Python*. Retrieved from https://plyer.readthedocs.io/
- Roesch, M. (1999). *Snort - Lightweight Intrusion Detection for Networks*. Proceedings of the 13th USENIX Conference on System Administration, 229–238.
- Gupta, B. B., Tewari, A., Jain, A. K., & Agrawal, D. P. (2016). *Fighting Against Phishing Attacks: State of the Art and Future Challenges*. Neural Computing and Applications, 28, 3629–3654.