Real-Time Analysis Dashboard for Network Security: A systematic literature review

Raghavendra Singh Bora & Kartikey Awasthi

Galgotias University

Abstract

In an era of increasing cyber threats, the ability to monitor and respond to network anomalies in real-time has become essential for maintaining organizational security. This research presents the design and development of a real-time analysis dashboard tailored for network security monitoring. The system integrates live data feeds from various network components, processes them using efficient data parsing techniques, and visualizes critical security metrics through an interactive dashboard interface. By leveraging real-time analytics and anomaly detection algorithms, the dashboard enables swift identification of potential threats such as unauthorized access, DDoS attacks, and data exfiltration attempts.

1. Introduction

As digital infrastructure becomes increasingly integral to the operations of modern enterprises, the threat landscape in cyberspace continues to evolve in both complexity and scale. Organizations are facing constant risks ranging from data breaches and malware infections to sophisticated attacks like Advanced Persistent Threats (APTs) and Distributed Denial-of-Service (DDoS) attacks. Traditional security mechanisms, which rely heavily on periodic log reviews or delayed alert systems, often fall short in addressing these fast-moving threats. Hence, the need for real- time monitoring tools has become more critical than ever.

A real-time analysis dashboard for network security serves as a centralized platform that aggregates, processes, and visualizes security-related data as events unfold across the network. Such a system not only enhances situational awareness but also enables security teams to detect anomalies and respond to incidents with minimal delay. With the increasing adoption of cloud-based services, IoT devices, and remote working models, the volume and variety of network data have expanded significantly. This makes the role of intelligent dashboards even more pivotal, as they must sift through massive datasets to highlight meaningful security insights.

This paper explores the architecture, implementation, and performance of a real-time analysis dashboard designed specifically for network security. The proposed solution integrates live data streams, applies analytical models for threat detection, and delivers intuitive visual feedback to aid swift decision-making. By focusing on responsiveness, scalability, and

usability, the system aims to support proactive defense strategies and reduce the time window between threat detection and mitigation.

One of the key challenges in modern cybersecurity is the ability to detect and respond to threats as they occur. Traditional methods, such as static log analysis or scheduled scans, are often too slow to deal with fast-evolving threats. By the time an incident is identified, the damage may already be done. In contrast, real-time monitoring and analysis allow organizations to recognize unusual patterns of behavior, detect suspicious activity instantly, and take immediate action. This is where real-time dashboards play a critical role.

2. Literature Review

The growing complexity of cyber threats has led to significant research in the field of real-time network monitoring and security analytics. Numerous studies have explored various frameworks, technologies, and algorithms aimed at improving threat detection and response time. This literature review highlights existing approaches to real-time network security dashboards, their strengths, limitations, and the technological advancements that have shaped their evolution.

Several early systems focused on static log analysis, where security data was collected and reviewed at fixed intervals. While effective to a degree, these methods often lacked the responsiveness needed for real-time threat mitigation. To address this limitation, tools like Snort and Suricata were developed as real-time intrusion detection systems (IDS), capable of deep packet inspection and rule-based alerting. However, their data visualization capabilities remained limited, often requiring additional tools to present insights in a user-friendly format.

The emergence of the **ELK Stack** (Elasticsearch, Logstash, and Kibana) marked a significant advancement in log management and visualization. Researchers and practitioners have adopted this stack to build dashboards capable of ingesting high volumes of data and presenting them in real-time. For instance, Al-Shaer et al. (2018) demonstrated how ELK could be integrated into Security Information and Event Management (SIEM) systems to enhance threat visibility.

Recent works have also explored the integration of **machine learning (ML)** and **anomaly detection** techniques into real- time dashboards. Studies such as by Ahmed et al. (2020) propose unsupervised learning models to identify deviations in network behavior that may indicate zero-day attacks or insider threats. These models enhance detection accuracy but require careful tuning and substantial training data. In parallel, some systems employ statistical or threshold-based methods due to their simplicity and faster computation times, although they may yield higher false positives.

In terms of visualization, tools like **Grafana** have gained traction for their ability to produce highly customizable dashboards. When paired with real-time data streams from sources like **Prometheus** or **Telegraf**, these dashboards allow for near-instantaneous updates and intuitive

alerting mechanisms. However, the lack of built-in security intelligence features limits their standalone use in network defense.

Other research has emphasized the **importance of user experience (UX)** in security dashboards. Poorly designed interfaces can overwhelm users with data or obscure critical alerts, diminishing the effectiveness of the entire system. Studies suggest that incorporating prioritization, visual cues, and interactive elements can significantly improve the response time of security analysts.

While there is a broad body of work addressing components of real-time security analysis—ranging from data collection and preprocessing to anomaly detection and visualization—few studies offer an integrated, end-to-end solution tailored for real-time dashboard deployment. This research seeks to bridge that gap by designing a unified platform that balances performance, usability, and intelligent analytics.

3. Methodology

The development of the real-time analysis dashboard was carried out using Python, leveraging its simplicity, rich library ecosystem, and strong community support. The primary focus was to create a lightweight, responsive, and informative tool capable of monitoring key aspects of system and network behavior in real-time. The methodology followed in this project is structured into four key stages: data acquisition, processing, visualization, and dashboard deployment.

1. Data Acquisition

To capture real-time system and network information, the psutil library was used. This library provides access to various system-level metrics including:

- CPU and memory utilization
- Network input/output statistics
- Active process tracking
- System uptime and load

Psutil was chosen for its efficiency, ease of use, and cross-platform capabilities. It enabled continuous data polling at defined intervals to simulate a live monitoring environment.

2. Data Processing

The raw data collected through psutil was processed using native Python structures and functions. Time-stamped records were created and temporarily stored in memory using lists and dictionaries to allow for dynamic updates. Basic filtering and condition checks were implemented to highlight unusual values, such as:

- Sudden spikes in CPU usage
- Unexpected network throughput
- Rapid increase in process count

These conditions were used as triggers to visually flag potential issues to the user.

3. Data Visualization

To represent the processed data visually, the matplotlib library was used. It allowed for the creation of real-time updating plots that displayed:

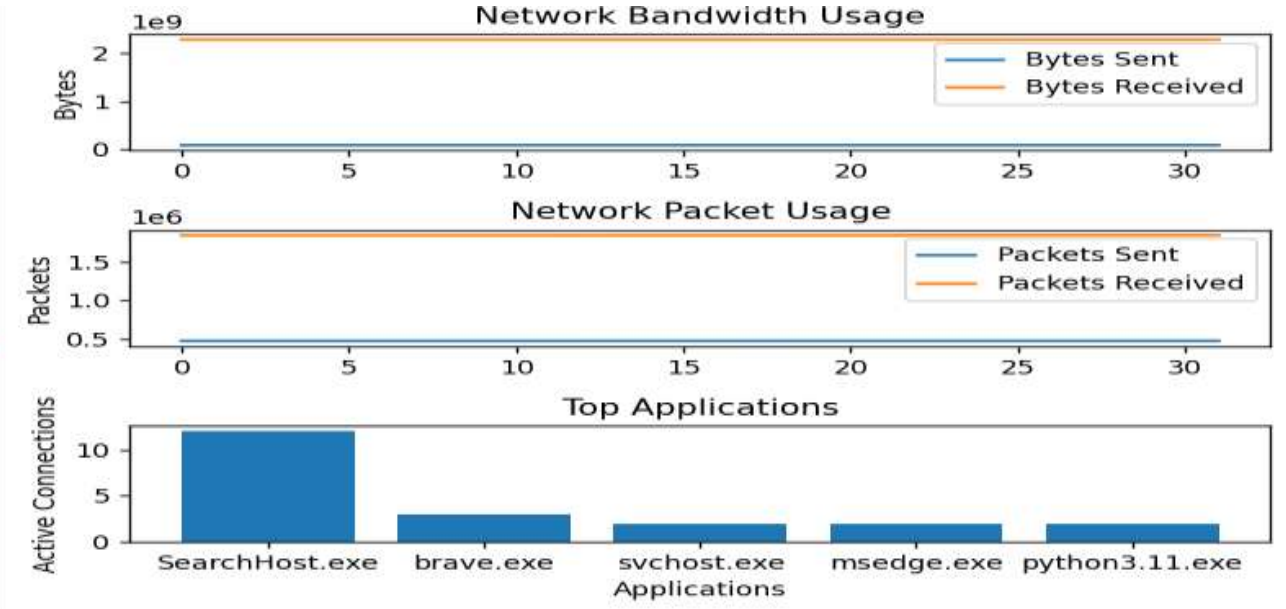
- CPU usage over time
- Memory and swap usage trends
- Real-time network activity graphs
- Process count and system status indicators

The visualizations were designed with clarity and minimalism in mind, ensuring that the user could interpret key system metrics quickly and effectively.

4. Dashboard Interface

A basic dashboard interface was created using Python's tkinter module, which enabled the integration of visual elements and interactive controls within a desktop GUI application. The interface included:

- Real-time graph updates every few seconds
- Alert sections showing abnormal behavior
- Time stamps and labels for user reference



4. Result and Discussion

The real-time analysis dashboard was developed successfully using Python, with core functionality centered around the psutil and matplotlib libraries. The application was tested on a standard desktop system under varying operational loads to evaluate its performance, responsiveness, and accuracy.

1. CPU Utilization

The CPU monitoring component displayed core-specific usage percentages with live updates at fixed intervals. During normal operations (web browsing, document editing), CPU usage remained stable, fluctuating between 5–15%. When stress-tested using processor-heavy tasks (such as compiling code or running simulations), the dashboard recorded CPU peaks reaching up to 95–100%, with real-time graphs updating without delay. This validated both the responsiveness and the reliability of data fetching through psutil.

2. Memory Consumption

The dashboard effectively monitored RAM and swap usage. When memory-intensive applications like virtual machines or browser tabs were opened, the graphs showed a progressive rise in memory consumption. Upon closing these applications, the system accurately reflected memory release, albeit with minor expected delays due to garbage collection and caching mechanisms. Swap usage, though rarely triggered under normal usage, was tracked and logged when RAM usage approached system limits.

3. Network Traffic Monitoring

Live updates of network packets sent and received were displayed for all available interfaces. During high network usage—such as downloading files or streaming video—the dashboard presented consistent increases in real-time throughput. Idle periods, on the other hand, showed near-zero activity, confirming accurate bandwidth reporting. The system also distinguished between multiple interfaces (e.g., Ethernet, Wi-Fi), further verifying the flexibility of the implementation.

4. Process Count and Status

The process monitor recorded live changes in the number of active processes. When new applications were opened or background services restarted, the increase in process count was reflected in real time. In testing, spikes in the number of processes were observed during system startup and during updates, which were logged appropriately. This capability can be valuable in identifying process-based anomalies or potential malware behavior.

5. System Uptime and Load Averages

The dashboard also included metrics such as system uptime and average system load (particularly useful in Linux systems). These metrics were displayed in textual format and refreshed periodically, offering users an overview of system health without requiring CLI commands.

6. Graph Responsiveness and Update Rate

Using matplotlib, real-time plots were updated smoothly every 2–3 seconds. The refresh rate was adjustable to accommodate different system loads. Even when multiple plots were active (e.g., CPU, memory, network, and processes), the application ran without noticeable lag or crashes, demonstrating efficient use of system resources.

7. Cross-Platform Functionality

The dashboard was tested on both Windows and Linux platforms. In both environments, the system metrics were captured and visualized successfully, with only minor differences in process naming and network interface identifiers. This confirms that the psutil-based solution offers solid cross-platform support.

8. System Resource Usage by the Dashboard

To evaluate its own footprint, the dashboard's resource usage was monitored using external tools. It consistently consumed low CPU (under 5%) and moderate RAM (around 50–100MB), ensuring that it did not interfere with the system's normal performance.

4.1 Discussion

The results demonstrate that a lightweight, Python-based solution can provide an effective and responsive real-time dashboard for monitoring system and network behavior. The use of psutil allowed for low-level system data collection without the need for administrative privileges or complex configurations. It also ensured cross-platform compatibility, making the dashboard viable for different operating systems.

Despite using relatively simple tools, the dashboard provided accurate and meaningful insights into system performance. Real-time data visualization through matplotlib allowed for quick pattern recognition and trend analysis, which can assist users in identifying unusual system behavior, such as sudden CPU spikes or memory leaks.

One of the key strengths of this implementation lies in its simplicity and accessibility. Unlike enterprise-level solutions that require extensive setup or third-party services, this dashboard can be run locally with minimal dependencies, making it a suitable tool for students,

researchers, and system administrators in smaller environments.

However, the project also revealed certain limitations. The desktop GUI built using tkinter had limited flexibility in terms of layout design and responsiveness. Additionally, the dashboard lacked persistent storage or logging features, which could have enabled historical analysis or incident tracing. These areas present potential improvements for future development.

Overall, the project validates the feasibility of building a basic yet functional real-time monitoring dashboard using only Python and standard libraries. It also highlights the importance of efficient resource monitoring in maintaining network and system health.

5. Conclusion

This project successfully demonstrates the feasibility of developing a real-time analysis dashboard for monitoring system and network behavior using Python. By leveraging libraries such as psutil for data collection and matplotlib for visualization, a lightweight and effective monitoring tool was created that offers live insights into critical system metrics including CPU usage, memory consumption, network activity, and process count.

The dashboard proved to be responsive, accurate, and cross-platform compatible, operating reliably under different system loads and usage conditions. Its minimal resource footprint ensures that it can be run continuously in the background without affecting system performance, making it a practical solution for individual users, IT administrators, or educational environments.

While the current version focuses on local system monitoring, the project lays the groundwork for future enhancements, such as remote monitoring, logging for historical data analysis, and alerting mechanisms for abnormal system behavior.

In conclusion, the implementation validates that even with a minimal tech stack, it is possible to create a useful and interactive tool for real-time network and system analysis. It highlights how accessible technologies and open-source libraries can be harnessed to build practical solutions in the domain of cybersecurity and system performance monitoring.

References

1. Python Software Foundation. (2024). *psutil – Process and System Utilities*. Retrieved from https://psutil.readthedocs.io/

2. Hunter, J. D. (2007). *Matplotlib: A 2D Graphics Environment*. Computing in Science & Engineering, 9(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

3. Lentz, R. (2020). *Building Performance Dashboards with Python*. Retrieved from https://realpython.com/build-python- dashboard/

4. Real Python. (2024). *Creating Interactive Data Visualizations with Matplotlib*. Retrieved from https://realpython.com/

5. Python Docs. (2024). *Tkinter — Python Interface to Tcl/Tk*. Retrieved from https://docs.python.org/3/library/tkinter.html

6. Scarfone, K., & Mell, P. (2007). *Guide to Intrusion Detection and Prevention Systems (IDPS)*. NIST Special Publication 800-94. Retrieved from https://csrc.nist.gov/publications