

# **Network Management System**

## **Automated Root Cause Analysis Platform**

By

B.Tech CSE-IOT & Cybersecurity Including Blockchain Technology  
7<sup>th</sup> Sem

Mohammed Maaz (R22EI042)  
Sahil Robert (R22EI049)

### **Abstract**

Modern enterprise networks generate thousands of alarms every day, most of which are secondary symptoms of a single underlying failure. Manually correlating these alarms leads to alert fatigue, slow troubleshooting, and increased Mean Time to Resolve (MTTR). This project presents an **Automated Root Cause Analysis (RCA) Platform** designed to eliminate manual intervention in identifying the primary cause of network incidents.

The work is executed in two phases. **Phase-0** studies an industry-standard monitoring architecture using Nagios, SNMP traps, and Apache Kafka for event collection, streaming, and alarm suppression. **Phase-1** introduces the actual implementation: a fully autonomous Python-based RCA Simulation Engine deployed on Ubuntu. This engine automatically injects failures, detects anomalies, correlates events, suppresses dependent alarms, determines the most probable root cause, and performs auto-remediation in real time.

By combining enterprise-grade concepts with a custom autonomous RCA engine, the system achieves faster incident detection, lower MTTR, reduced operational noise, and improved network reliability—demonstrating an efficient, open-source approach to automated network troubleshooting.

## Background Studies

### 1. Current Problem Statement

Large enterprise networks constantly generate huge volumes of alerts—link failures, high CPU usage, packet drops, service outages, and dependency failures. In most cases, **a single root cause** triggers **multiple downstream alarms**.

However, operators must manually:

- Inspect raw alarms
- Correlate timestamps
- Analyze dependency relationships
- Identify the actual root cause

This manual process leads to:

- **Alert fatigue**
- **High MTTD (Mean Time to Detect)**
- **High MTTR (Mean Time to Resolve)**
- **Delayed decision-making**
- **Increased operational workload**

As networks scale, manual troubleshooting becomes impractical. A system that can **automatically correlate alarms and determine the root cause** is essential.

### 2. Existing Solutions

Industry solutions such as **IBM Netcool**, **SolarWinds**, and **MicroFocus OMi** provide automated event correlation and RCA.

However, these platforms are:

- Expensive
- Closed-source
- Difficult to customize
- Heavyweight for academic or small-scale deployments

Open-source tools like **Nagios**, **Zabbix**, and **Prometheus** focus primarily on monitoring. While they excel at alerting and performance tracking, they **lack native RCA**, intelligent correlation, and automatic alarm suppression.

This gap highlights the need for a **lightweight, open-source, automated RCA mechanism**.

## Proposed Objectives

1. **To design and implement an Autonomous Root Cause Analysis Engine** capable of functioning without Nagios/Kafka.
2. **To simulate real-time network failures automatically**, including latency spikes, service outages, and resource saturation.
3. **To detect anomalies in real-time** using threshold-based and deviation-based monitoring of system metrics.
4. **To perform automated event correlation** using dependency graphs and fault propagation.
5. **To identify the most probable root cause** using ranking logic and incident clustering.
6. **To implement automatic alarm suppression**, eliminating redundant and dependent alerts.
7. **To perform self-remediation**, restoring system stability without operator intervention.
8. **To track incident lifecycle metrics** such as MTTD, MTTR, stability score, and system recovery time.
9. **To deploy the complete RCA engine on Ubuntu**, ensuring portability, scalability, and open-source compatibility.

## Proposed Methodology

The objective of this project is to design and implement an **Automated Root Cause Analysis (RCA) Platform** capable of identifying, correlating, and diagnosing system incidents without human intervention. The goal is to eliminate manual log-checking and alarm correlation by integrating **Nagios event handlers, SNMP trap processing, and Kafka-based event streaming** to build a real-time, self-correcting monitoring system. The platform aims to reduce troubleshooting time, eliminate false alarms through suppression logic, and provide accurate, immediate insights into the actual cause of each incident.

### Step-by-step Methodology:

1. **Network Monitoring Setup (Nagios Core):**  
Nagios monitors servers, routers, and switches using service checks and

SNMP traps. Each network device sends event data (up/down status, performance metrics) to Nagios.

**2. Event Collection via SNMP Traps:**

SNMP traps are configured on devices to send alerts directly to the Nagios SNMP Trap Daemon (snmptrapd). These traps carry critical fault information that gets logged and parsed into Nagios.

**3. Event Streaming with Kafka:**

To handle real-time scalability, each event from Nagios (or SNMP) is pushed into an Apache Kafka topic. Kafka acts as the intermediary message broker, capable of processing thousands of alerts per second and distributing them to RCA modules.

**4. RCA and Alarm Suppression Logic:**

Using Nagios **event handlers**, each incoming alert is analyzed for dependency relationships. If multiple devices fail due to a parent node (e.g., router down causes downstream switches to fail), the handler suppresses child alarms and marks the parent as the root cause.

The event correlation logic works as follows:

- Fetch dependent service/host relationships from Nagios configuration.
- Identify common failure nodes based on timestamp and dependency mapping.
- Suppress repetitive or dependent alarms to avoid alert flooding.
- Update Nagios web UI and Kafka event log with RCA results.

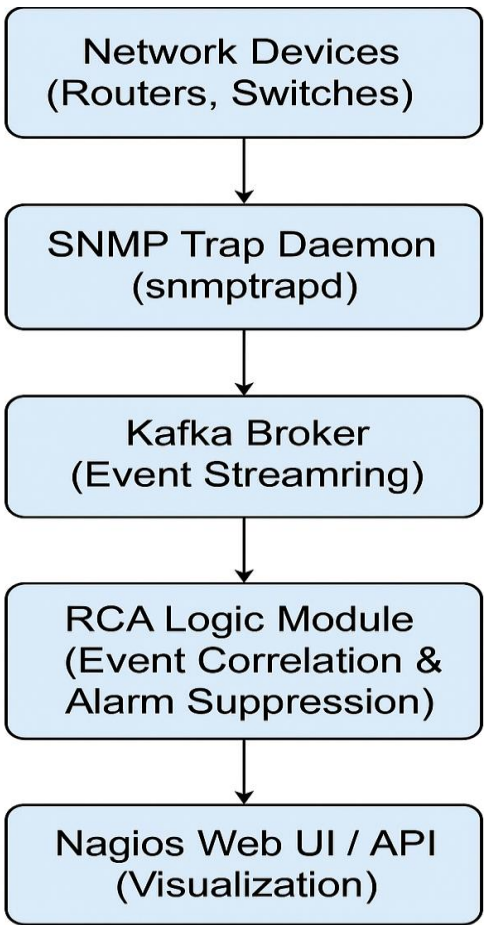
**5. Visualization and Alerts:**

The final RCA results are displayed on the Nagios web dashboard and can trigger email/SMS/Slack notifications. Operators see only root-cause alerts, reducing noise and response time.

**6. Testing and Validation:**

The platform is tested by simulating multiple network failures. Metrics such as the number of suppressed alarms, RCA accuracy, and response time are evaluated.

Flow Diagram



Software Stack and details

Software Name	Version	Status (Open Access or Licensed or Trail version)	Link
VMware Workstation Pro	252H	Licensed (Free for non-	<a href="https://www.vmware.com/products/workstation-player.html">https://www.vmware.com/products/workstation-player.html</a>

		commercial use)	
Ubuntu Linux	24.04 LTS	Open Access	<a href="https://ubuntu.com/download">https://ubuntu.com/download</a>
Nagios Core	4.4.14	Open Access	<a href="https://www.nagios.org/downloads/">https://www.nagios.org/downloads/</a>
Apache Kafka	3.9.0	Open Access	<a href="https://kafka.apache.org/">https://kafka.apache.org/</a>
SNMP & SNMP Trap Daemon	5.9.3	Open Access	<a href="https://net-snmp.sourceforge.io/">https://net-snmp.sourceforge.io/</a>
Apache HTTP Server	2.4.x	Open Access	<a href="https://httpd.apache.org/">https://httpd.apache.org/</a>
PHP	8.x	Open Access	<a href="https://www.php.net/">https://www.php.net/</a>

### Expected Outcome

1. Automated Root Cause Analysis of network incidents using alarm suppression and event correlation.
2. Significant reduction in operator workload and faster troubleshooting with fewer redundant alerts.