# Open-Source Tooling in Cloud Environments for Lawful Interception Tools

Borja Marazuela
School of Enterprise Computing and Digital Transformation, TU Dublin, Ireland
X00218724@myTUDublin.ie

## Introduction

This research addresses the need for cost-effective and transparent Lawful Interception solutions for network operators shifting to cloud environments. By establishing quantitative benchmarks for OpenLI in Google Cloud Platform (GCP), the study compares three deployment models: a single virtual machine (VM), a distributed architecture with dedicated VMs per service, and a containerized environment managed via Kubernetes. Performance was measured using synthetic VoIP traffic loads to evaluate latency, throughput, and packet loss.
Our results indicate that OpenLI's performance scales linearly with increased CPU cores, though it is currently bottlenecked by its single-instance mediation function and the capacity of Law Enforcement Agency (LEA) systems. Ultimately, our testing found the distributed VM-per-service model to be the most efficient and performant solution. It offers superior flexibility for resource optimization without the added complexity and architectural friction introduced by container orchestration tools.

## Lawful interception

Lawful Interception (LI) is formally defined by the ITU as the "lawfully authorized interception and monitoring of telecommunications pursuant to an order of a government body" to obtain forensic evidence against wrongdoers. It represents the technical and legal intersection of national security needs—such as investigating organized crime and terrorism—and the societal right to privacy. Modern LI has evolved from physical wiretapping of copper lines to the complex, automated capture of data in packet-switched networks, governed by frameworks like CALEA in the US and ETSI standards internationally.

## State of the art

Current research in the LI field focuses on modernizing legacy systems through several key trends: Lawful Interception as a Service (LiaaS), which leverages cloud-native technologies for media analysis and intelligent extraction; Self-Sovereign Identity (SSI) to automate and secure interactions between agencies; and specialized mechanisms for 5G networks to address end-to-end encryption challenges. OpenLI itself represents a major step forward as an ETSI-compliant open-source alternative that lowers entry barriers for network operators by reducing reliance on expensive, proprietary hardware

## Empirical testing

**1. OpenLI Component architecture:**

OpenLI utilizes a microservices-based architecture consisting of three primary components:

- Provisioner: Acts as the administrative interface, receiving warrants manually through a web portal or automatically via an API and translating them for the network.
- Collector: Mirrors network traffic and filters for packets belonging to intercepted targets based on the list provided by the provisioner.
- Mediator: Encompasses mediation and delivery, adding ETSI-compliant headers and buffering data in a RabbitMQ queue for delivery to the Law Enforcement Monitoring Facility (LEMF).

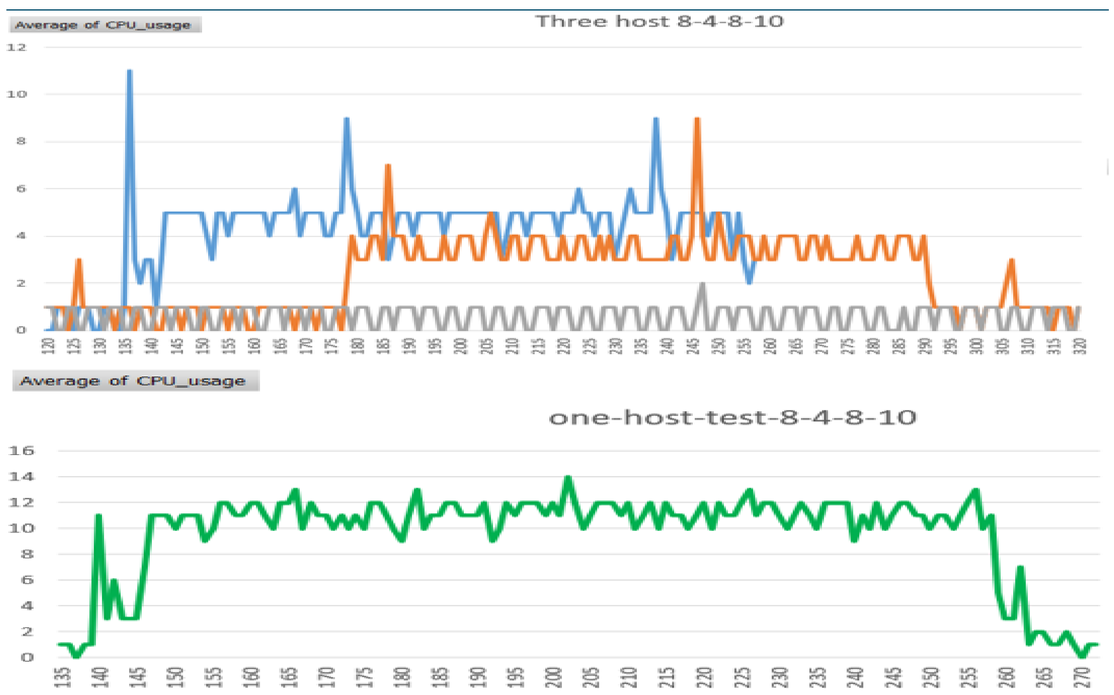
**2. VM environment:**

Several VMs were tested with various configurations. For Scenario 1 (single VM), performance scaled near-linearly with CPU cores, with 2 vCPUs being the minimum required to prevent packet loss. Scenario 2 (VM-per-service) demonstrated that the Collector is the primary CPU consumer during processing, while the Mediator uses more memory to buffer retransmissions.
Stress testing revealed the system's limits; at this peak, the system began dropping packets because the collector could no longer handle the incoming traffic volume. There was no significant difference in total resource usage between the single-host and three-host VM setups.

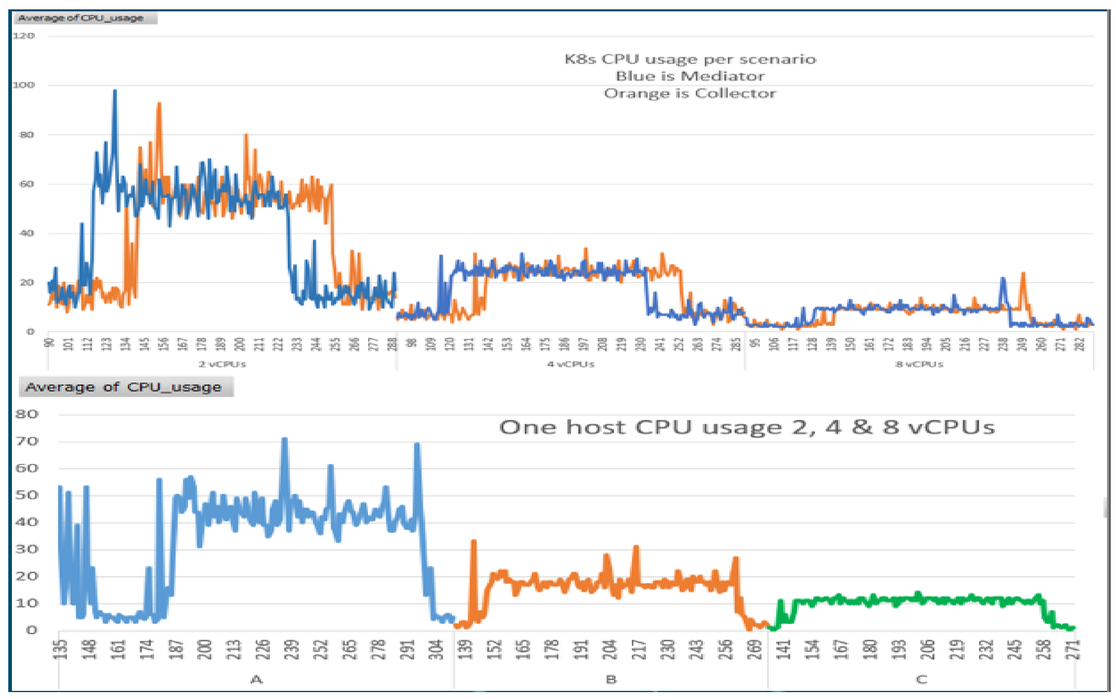
**3. Kubernetes environment:**

Scenario 3 utilized Google Kubernetes Engine (GKE) to deploy the three OpenLI services in individual pods, with scaling tested across different host configurations. While GKE provided high availability and self-healing, the deployment process was more complex due to the need to reconfigure local RabbitMQ instances upon every restart. In the 2 vCPU scenario, the system was at capacity suggesting higher overhead than the VM scenarios.
GKE results mirrored the VM performance in terms of linear scaling, but the environment introduced resource under-utilization due to the required parity between cores and RAM.

## Graphic Results




(a) Host per service CPU usage comparison




(b) Kubernetes CPU usage comparison

## Conclusions and Future Work

Our testing shows that separated VMs are currently the most practical and performant deployment method for OpenLI. This model allows for granular resource allocation. Future research lines are enhancing OpenLI by adding extra features and replicating this testing on other cloud providers.

## QR Code for Recording