March 2025

Demonstrating the Feasibility of a Containerized Security Platform: A Technical Proof of Concept

Ozan Ayik
Goldoak GmbH
Security Engineer

Abstract

This proof of concept (PoC) demonstrates the technical feasibility of a containerised, microservices-based security platform designed for small and medium-sized enterprises (SMEs). Using Docker and Kubernetes, we have deployed several Python services capable of real-time monitoring and security analysis across 15 systems in our test environment. Over a continuous two-month period, the platform operated stably and maintained real-time visibility of endpoint activity without significant resource overhead. These initial results highlight our lightweight architectural approach and confirm the system's suitability for micro and small businesses seeking enterprise-level security on a modest infrastructure. While current data is limited to our own environment, future PoCs and pilot deployments will focus on scaling to larger customer bases and refining automated threat detection capabilities.

Contents

List of Figures

List of Tables

Listings

1. Introduction

1.1. Introduction

This Proof of Concept (PoC) document has been prepared primarily for Innosuisse reviewers, potential investors and other stakeholders interested in assessing the technical feasibility of our containerized cybersecurity platform called “Goldoak Spectacles”. By presenting concrete implementation details and initial performance observations, we aim to demonstrate that our core architecture can efficiently address key security needs - especially in small and medium-sized enterprises (SMEs).

1.2. Scope and Objectives

The scope of this PoC is deliberately focused on validating the basic design and operational viability of our platform, rather than exhaustively testing every possible feature. Specifically, we set out to:

Demonstrate Containerized Microservices: Show how our Python-based services can be deployed using Docker and orchestrated with Kubernetes to achieve a lightweight, scalable foundation for cyber defense.

Highlight Core Security Functions: Provide evidence that the platform can perform real-time monitoring, automated vulnerability scanning, and basic logging/analysis—even at a modest scale of up to 15 systems.

Evaluate Stability Under Limited Resources: Assess whether the architecture remains stable and efficient under minimal infrastructure, reflecting the practical constraints of SMEs.

These objectives guide the tests and demonstrations within this PoC. Advanced or planned functionalities (such as AI-driven threat intelligence or extended compliance features) remain out of scope for now, although we discuss them briefly in the concluding Roadmap section.

1.2.1. Overview of the Platform

Our platform is designed as a **Next-Generation Security Operations Center (SOC)** solution, leveraging a containerized microservices architecture to deliver **enterprise-class** capabilities - such as endpoint protection, vulnerability management, and continuous monitoring - at a cost structure and deployment scale suitable for SMEs. Each microservice (written in Python) addresses a specific topic, such as vulnerability management, event processing, or open source intelligence (OSINT). These services run in **Docker containers**, which are then orchestrated via **Kubernetes** for auto-scaling, load balancing, and fault tolerance.

Data ingestion and analysis is based on a combination of lightweight collectors and a centralized data store (Elastic Stack), enabling rapid search and correlation of security events. By decoupling functions into separate microservices, we ensure that each component can be updated, scaled, or replaced independently, in line with modern DevSecOps principles. This flexible architecture not only supports rapid deployment across multiple environments (on-premises, cloud, or hybrid), but also paves the way for future enhancements-such as adding AI-driven anomaly detection modules or integrating third-party threat intelligence feeds.

In the following sections, we describe the system architecture in more detail, explain the key steps we took to implement and test our PoC, and present preliminary results that underscore the platform's readiness for further pilot deployments.

2. System Architecture

2.1. High-Level Diagram

In this PoC, we deploy our core security platform on a single-node Kubernetes cluster (minikube) running on a Debian server. As shown in Figure 2.1, the primary node hosts several containerized services, ranging from our Goldoak backend and Goldoak front-end to Elasticsearch and Kibana, all orchestrated under Kubernetes. A separate Docker-based application, located on the same or a different Debian server, runs additional containers (such as the Goldoak Vulnerability Worker and an OpenVAS container) to perform more specialized security tasks.

In Figure 1, the left box highlights the main Minikube cluster on a Debian server, which hosts our main microservices - backend, frontend, OSINT module - and the Elastic Stack components (Elasticsearch, Kibana). An Nginx ingress handles the internal routing for these containerized services. Meanwhile, the right box shows a separate Docker application running our vulnerability worker, which interacts with an OpenVAS container. This worker coordinates the scanning process, parses the data and shares the results with the Goldoak backend via a RESTful API call. The architecture of the platform is thus both modular and extensible, allowing additional security modules or external tools to be integrated with minimal configuration changes.

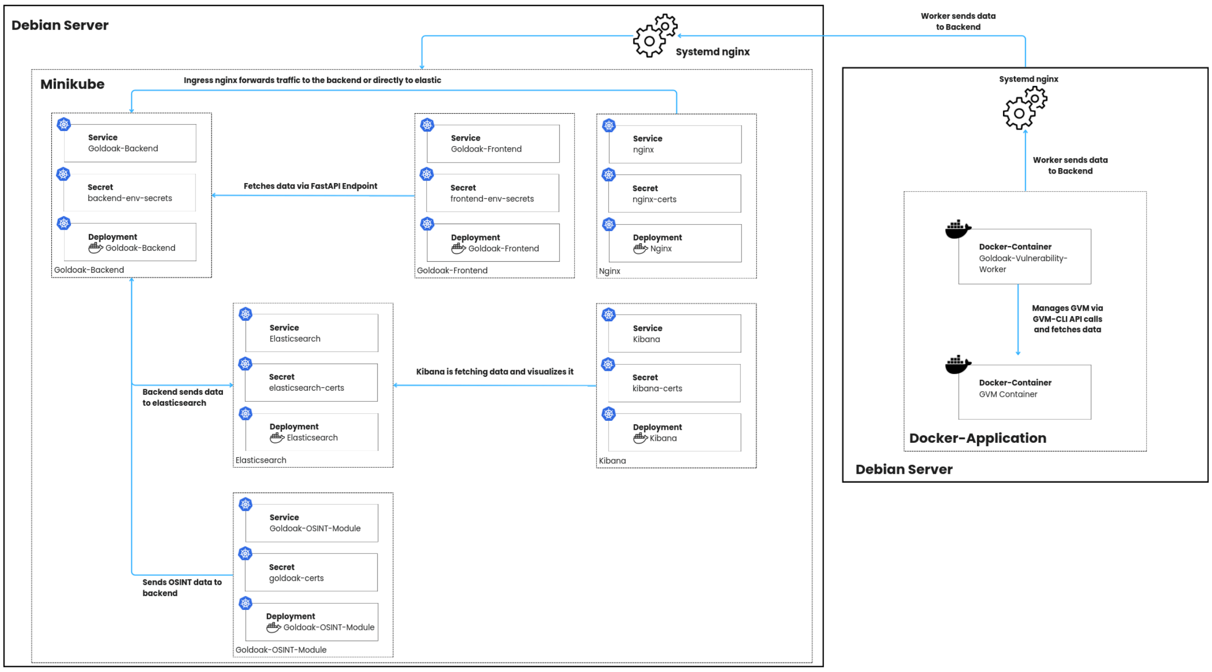


Figure 1: High level overview of our implementation showing our main application “Goldoak Spectacles”.

2.2. Key Components and Their Interactions

Table 1: Key Components and Interactions - Page 1

Component	**Description**
Goldoak-Backend (Python FastAPI)	• Service & Deployment: Hosted in a Kubernetes deployment, exposing a RESTful API for data management (e.g., OSINT results, vulnerability findings). • Credentials: Manages environment-specific credentials and certificates (e.g., for secure communications, database credentials). • Data Flow: Receives data from both the OSINT module and the external Vulnerability Worker, then passes relevant information to Elasticsearch for indexing.
Goldoak-Frontend (UI)	• Service & Deployment: Runs inside Kubernetes, providing users with a high-level view of KPIs. • Purpose: Displays aggregated results from Elasticsearch (via the backend) such as discovered vulnerabilities, an asset inventory, and OSINT findings. • Secrets: Stores environment variables (e.g., API endpoints, credentials) securely.
Goldoak-OSINT-Module	• Service & Deployment: Dedicated Python microservice that retrieves public information about specified servers or IP addresses. • Data Flow: Sends OSINT data to the Goldoak-Backend, which in turn indexes the data in Elasticsearch. • Benefit: Allows real-time checks of open-source intelligence (e.g., domain info, leaked credentials) to alert on potential security exposures.

Table 2: Key Components and Interactions - Page 2

Component	**Description**
Elasticsearch & Kibana	• Elasticsearch: Collects, indexes, and stores large volumes of security-related data and makes it rapidly searchable. • Kibana: Provides visualization dashboards and query capabilities. In this PoC, it is used to create real-time views of vulnerabilities, OSINT results, and system logs. • Secrets: Manages certificates for secure communications (e.g., TLS).
Nginx (Ingress)	• Role: Acts as an Ingress controller within Kubernetes, routing traffic either to the Goldoak backend or directly to Elasticsearch. • Security: Maintains TLS termination and load balancing rules for incoming requests, ensuring secure external access to internal services.
Vulnerability Worker & OpenVAS	• Location: Installed on an external server, outside the Minikube cluster. • Vulnerability worker: A specialized container that manages scans via OpenVAS (GVM-CLI calls), aggregates the results, and sends the processed data back to the Goldoak backend via a secure HTTP endpoint. • OpenVAS: An open source vulnerability scanner container that provides detailed scan results for systems in scope.

2.3. Technologies and Frameworks Used

Table 3: Technologies and Frameworks Used

Component	Description
Kubernetes (Minikube)	• Provides container orchestration, service discovery, and scaling. • Simplifies deployment in a single-node environment that mirrors real-world cluster functionality.
Docker	• Containerizes each microservice and the external vulnerability scanning components. • Ensures reproducible environments and simplifies deployment.
Python FastAPI	• Back-end framework chosen for its lightweight, asynchronous capabilities and straightforward development workflow. • Enables rapid creation of REST endpoints for integration with various security modules
Elastic Stack (Elasticsearch & Kibana)	• Enables storage and visualization of security data, logs, and findings in near real-time. • Scales from small PoC deployments (like this one) to enterprise-grade workloads.
Nginx	• Serves as both an Ingress Controller in Kubernetes and a systemd-managed service. • Routes traffic securely to backend services or Elasticsearch.
OpenVAS (with Docker)	• Provides an open-source vulnerability scanning engine. • Integrates with the Goldoak-Vulnerability-Worker for automated scan workflows.

This architectural layout demonstrates how multiple security-focused microservices can be orchestrated on a single node for demonstration purposes, while also showing how external Docker containers can extend or complement existing capabilities. In a real production environment, this design scales to multi-node Kubernetes clusters, additional vulnerability scanners, and more modules such as threat intelligence, configuration management, and more.

3. Why Elasticsearch & Kubernetes: Strategic and Technical Justifications

In designing our platform, we evaluated several approaches for deploying, scaling, and managing security-focused services. Ultimately, we chose Kubernetes as our container orchestration system and a microservices model for service decomposition. This decision was motivated by both industry case studies and academic literature, which consistently highlight the benefits of microservices and Kubernetes for modern cloud or on-premises deployments.

In addition, we evaluated several storage and query solutions when designing our data ingestion and analysis pipeline. Ultimately, we selected Elasticsearch to serve as our core data store for security events, logs, and performance analytics. Below are the key reasons for this decision, supported by both industry case studies and academic papers:

3.1. Industry Case Studies for Kubernetes

Tempestive's Nuboj (IoT Use Case): Tempestive, a software company specializing in manufacturing systems integration, revamped its IoT product, Nuboj, by adopting microservices in a Kubernetes environment. The result was a dramatic reduction in operational costs, the flexibility to support both cloud-based and on-premises models, and the ability to handle tens of thousands of devices while seamlessly scaling to hundreds of thousands. Crucially, the move to microservices has enabled the company to modularize and integrate proprietary functionality more efficiently - demonstrating real-world viability for small or mid-sized deployments [1].

Deltatre's GitOps Transformation: Deltatre, a leader in sports broadcasting and graphics, turned to Kubernetes, Helm, and Flux to streamline automation and scale across multi-cloud environments. By adopting these tools, they achieved a 50% reduction in pipeline runtimes and simplified onboarding for repeatable development environments. This underscores how container orchestration not only improves performance and scalability, but also significantly reduces operational overhead [2].

Grafana Labs (Vulnerability Scanning): Grafana Labs built an event-driven architecture on top of Kubernetes, using Dapr for inter-service communication and pub/sub mechanics. By containerizing each part of their vulnerability scanning pipeline, they gained resiliency (via automated retries and statestore-based idempotency) and easier scaling when dealing with tens of thousands of containers. This case highlights how microservices in Kubernetes can reliably handle bursty workloads, an important feature for real-time security operations [3].

CNCF Annual Survey (2023): According to the Cloud Native Computing Foundation, 84% of organizations surveyed are using or evaluating Kubernetes, indicating that it has effectively become the global standard for container orchestration. The survey also shows that only 15% of organizations have no current plans for Kubernetes, illustrating the critical mass it has reached across the industry [4].

3.2. Industry Case Studies for Elasticsearch

EY (Generative AI for Finance): EY adopted Elasticsearch to power a generative AI solution that helps financial institutions navigate large volumes of unstructured documents. By integrating with Elastic's Retrieval Augmented Generation (RAG), they achieved 10–15% accuracy gains and reduced processing time, demonstrating Elasticsearch's strengths in high-speed indexing and AI-driven workflows [5].

AHEAD (Managed SOC): AHEAD uses Elastic Security to ingest and analyze billions of monthly events for its MSSP business. Thanks to Elasticsearch's machine learning features and scalable data ingestion, they cut triage times by 73% and maintained a 6.9-minute mean time to response, even as security events rose by 50%. This underscores Elasticsearch's real-world effectiveness in high-volume security environments [6].

Global Pharmaceutical Corporation (R&D and Compliance): A Fortune 100 pharmaceutical enterprise stores over 95 terabytes of research data in Elasticsearch, enabling near-instantaneous retrieval and drastically simplifying regulatory reporting. By indexing millions of files with metadata, scientists reduced research lookups from days to seconds, improving collaboration and compliance simultaneously [7].

3.3. Academic and Professional Insights

Pahl (2015) shows that moving from virtual machines to containers for microservices provides significant benefits in portability, performance, and management. While this paper predates the highest wave of Kubernetes adoption, it laid important groundwork for container-based architectures [8].

Oyeniran et al. (2024) provide a comprehensive overview of microservices design patterns-such as Circuit Breaker, Service Discovery, and Strangler Fig-and emphasize the benefits of horizontal scaling and fault isolation when adopting microservices. The authors cite improved resilience, faster development cycles, and easier domain-driven modularization as key reasons to abandon monolithic approaches [9].

In terms of Elasticsearch specifically, Tiwari and Mane demonstrate how the ELK Stack can offer near-real-time log ingestion and dashboards for application performance monitoring, significantly streamlining troubleshooting efforts [10]. Meanwhile, Voit et al. highlight Elasticsearch's ability to manage massive data volumes in a distributed manner, enabling advanced full-text searches and visualizations for security, user behavior, and Big Data use cases [11].

4. Implementation Details

4.1. Containerization & Orchestration (Kubernetes, Docker)

Our platform is containerized using Docker and deployed in a Kubernetes environment. For this proof of concept (PoC), we chose a single-node minikube setup on a Debian server to closely mirror real-world orchestration without the complexity of a multi-node cluster. This approach ensures a lightweight yet production-like environment that can scale with minimal changes in the future.

4.2. Directory Structure & Deployment Process

Below is an excerpt of our minikube repository layout:

```Bash
.
├── agent-manifest.yml
├── bin
│   ├── generate-certs.sh
│   ├── install-app.sh
│   └── rebuild.sh
├── charts
├── elasti-agent.md
├── kubectl
├── manifests
│   ├── configmaps/
│   ├── deployments/
│   ├── ingress/
│   ├── namespaces/
│   ├── pvc/
│   ├── secrets/
│   └── services/
├── pki/
│   ├── ca/
│   └── issued/
├── README.md
└── ...
```

Listing 1: Minikube repository layout

The `manifests/` directory contains our primary Kubernetes YAML files, including:

- **Deployments** (e.g., `backend-deployment.yaml`, `elasticsearch-statefulset.yaml`)
- **Ingress** definitions for routes and load balancing (`nginx-ingress.yaml`, etc.)
- **ConfigMaps** for storing non-secret environment variables
- **Services** exposing each component internally or externally

- **PersistentVolumeClaims** for Elasticsearch, Kibana, and other stateful components

In the `bin/` folder, we store utility scripts:

- **generate-certs.sh**: Automates creation of self-signed TLS certificates for internal pod-to-pod encryption.
- **install-app.sh**: A quick-install procedure that applies all relevant Kubernetes manifests.
- **rebuild.sh**: Rebuilds and redeploys Docker images upon code changes.

4.2.1. Single-Node vs. Multi-Node Flexibility

While our PoC runs on Minikube (a single-node cluster), the same YAML definitions (under `manifests/`) can be deployed on a multi-node Kubernetes cluster with little modification. This ensures that as the platform grows or requires higher availability, we can migrate to a more robust environment (such as a managed K8s service or an on-premises cluster) without refactoring the core architecture.

4.2.2. Security & Secrets Management

We secure sensitive credentials, such as elasticsearch passwords and TLS keys, by using Kubernetes **Secrets** instead of transmitting them in cleartext. For inter-service communication: - **TLS certificates**: A local PKI (under `pki/`) issues certificates for Elasticsearch, Kibana, Nginx ingress, etc.

- **Ingress Nginx**: Terminates external TLS connections and redirects traffic to the correct services.
- **Pod-to-Pod Encryption**: Allows internal microservices to communicate over HTTPS to minimize the risk of eavesdropping.

Together, these measures ensure that our platform adheres to security best practices from certificate generation to intra-cluster communication.

4.3. Microservices Architecture (Python Services)

Our application follows a microservices architecture, with each service running in its own Docker container and communicating via HTTP(S) or message-based APIs. This modular design allows us to scale or update components independently, which aligns with DevSecOps practices for continuous integration and deployment.

4.3.1. Current Microservices

We currently maintain three core microservices, all written in Python and containerized with Docker. Depending on the deployment scenario, these containers can run either as standalone Docker services or within the Kubernetes cluster:

1. **Goldoak-OSINT-Worker**

  - **Purpose**: Gathers open-source intelligence (OSINT) about IP addresses or other identifiers (e.g., email domains). It discovers any exposed or leaked information relevant to the target, helping detect potential shadow IT or security threats.

```bash
.
├── app
│   ├── connector/
│   ├── main.py
│   └── workers/
│       └── osint_worker.py
├── bin/
├── Dockerfile
└── requirements.txt
```

Listing 2: simplified directory structure of the Goldoak-OSINT-Worker

- **Expansion**: Additional OSINT tools or scanning capabilities can be added simply by introducing new Python files under the `workers/` directory, maintaining a clean, extensible code base.

2. **Goldoak-Vulnerability-Worker**
   - **Purpose**: Automates vulnerability scanning (e.g., via Greenbone or other scanning engines), parses results, and ingests data into Elasticsearch. This provides end-to-end coverage from target definition to final report visualization.

```bash
.
├── app
│   ├── api/
│   ├── modules/
│   │   └── greenbone/
│   └── main.py
├── bin/
├── Dockerfile
└── requirements.txt
```

Listing 3: Simplified directory structure of the Goldoak-Vulnerability-Worker

- **Workflow**:
  1. **Define Targets**: Accepts IP ranges or hostnames via REST.
  2. **Scan & Parse**: Launches scans, retrieves XML or JSON reports, and parses them using `gvm_parser.py` or similar utilities.

3. **Ingest**: Uploads parsed data to Elasticsearch (or the Goldoak-Backend), enabling immediate correlation, alerting, or dashboarding.

3. **Goldoak-Backend**
   - **Purpose**: Acts as the central coordination point for other microservices. Handles user requests, job scheduling, and data aggregation.
   - **Integration**: Both the OSINT-Worker and Vulnerability-Worker communicate with this backend to share results, trigger tasks, or fetch new scanning directives.

4.3.2. Communication & Data Flows

Microservices primarily communicate via:

- **HTTP REST Endpoints** (exposed by the Goldoak-Backend or Worker routers).
- **In-Cluster Networking** if deployed to Kubernetes (Service objects route traffic using stable DNS names).
- **Secure Connections**: TLS is enforced on critical endpoints, leveraging certificates provisioned by our internal PKI or Let's Encrypt (for external traffic).

4.3.3. Future Expansion

As our platform evolves, we can introduce additional workers (such as a dedicated threat intelligence module or configuration compliance scanner) by simply creating new Docker services and plugging them into the backend. This approach preserves loose coupling and ensures that we can extend functionality without rewriting existing modules.

By adhering to the principles of microservices-small, focused components that communicate via well-defined APIs-we maintain clear lines of responsibility, which facilitates both development agility and robust fault isolation.

5. Functionality Demonstration

This section demonstrates how our platform works in practice, from initial deployment to performing key security and monitoring tasks. We provide screenshots, and links to short demo videos or GIFs that illustrate the real-world feasibility and user experience.

5.1. Deployment Steps (Quick Start)

Figure 2 illustrates a concise workflow for spinning up our solution on a single-node Kubernetes (minikube) environment. The installation is fairly straightforward, allowing us to deploy our solution on a system with a single script.

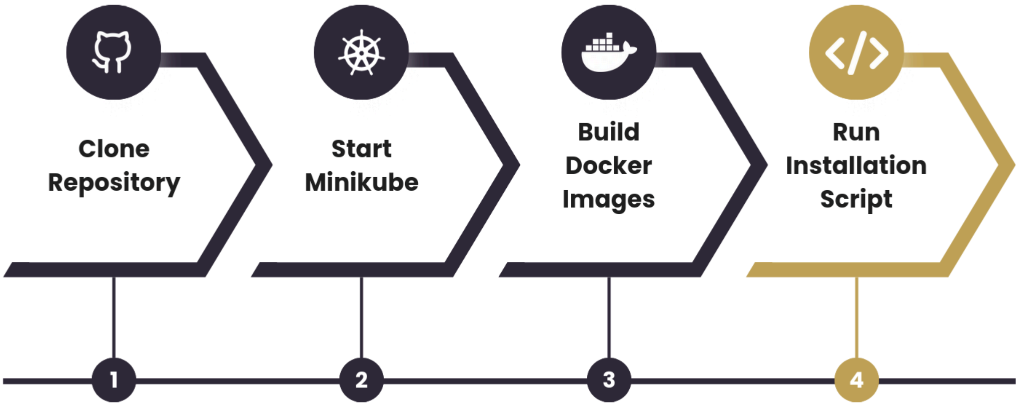


Figure 2: Deployment Steps (Quick Start)

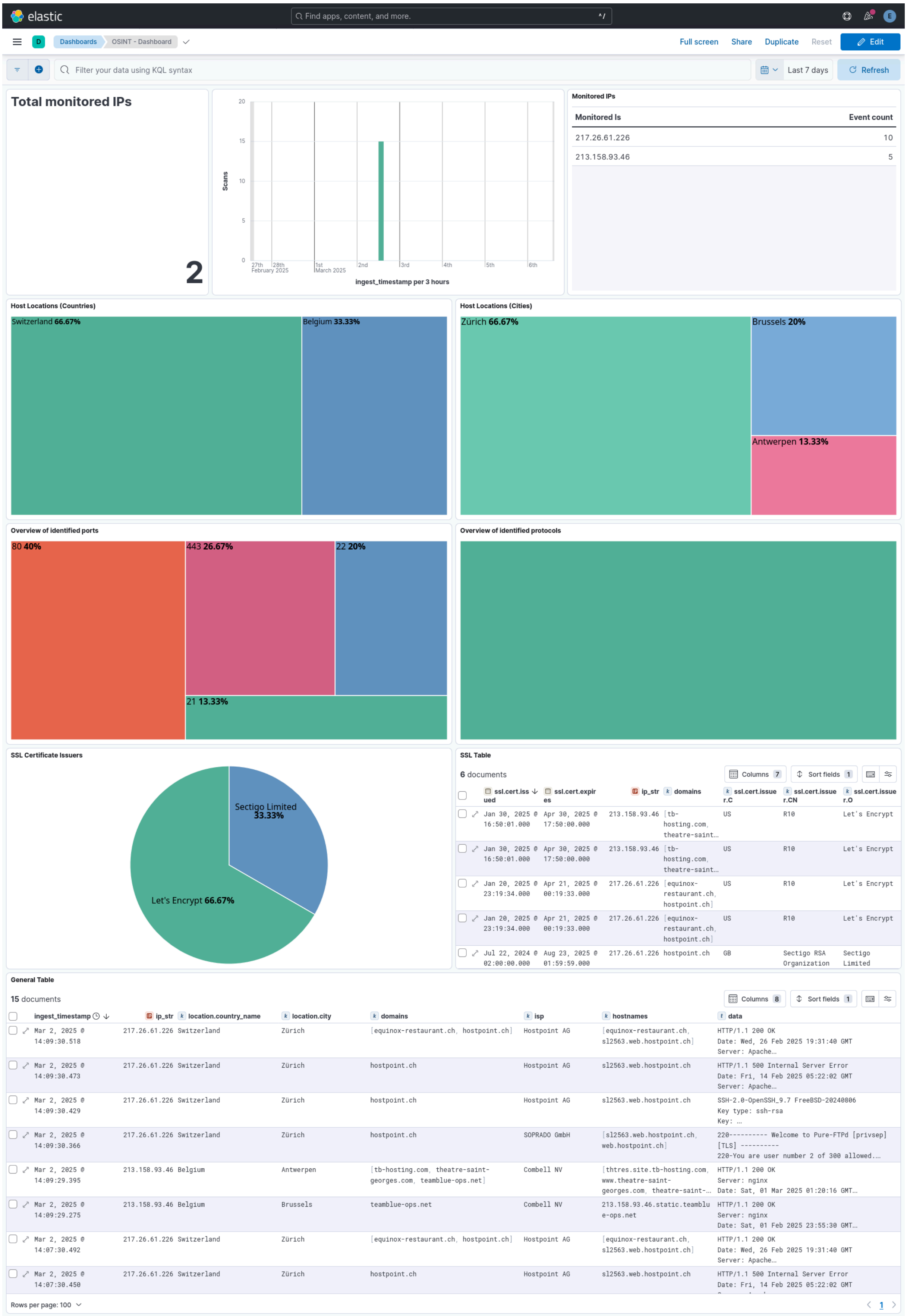
5.2. Core Features Demonstrated

5.2.1. OSINT Monitoring

Our platform's OSINT dashboard (see Figure 3) provides a real-time, visual representation of all publicly available information collected on monitored IP addresses or domains. After the Goldoak OSINT worker collects data from various sources, the results are indexed in Elasticsearch and automatically presented in Kibana.

Use Case:

- **Detect Shadow IT**: Unexpected domains or countries can signal potentially rogue infrastructure.
- **SSL Health Monitoring**: Detecting soon-to-expire certificates or unknown CAs ensures compliance with corporate standards.
- **Exposed Services Investigation**: Identifying open ports or suspicious server banners (such as outdated FTP services) prompts timely security action.



Monitored IPs

Monitored Is	Event count
217.26.61.226	10
213.158.93.46	5

SSL Table

6 documents

ssl.cert.issued	ssl.cert.expires	ip_str	domains	ssl.cert.issuer.C	ssl.cert.issuer.CN	ssl.cert.issuer.O
Jan 30, 2025 @ 16:50:01.000	Apr 30, 2025 @ 17:50:00.000	213.158.93.46	[tb-hosting.com, theatre-saint…	US	R10	Let's Encrypt
Jan 30, 2025 @ 16:50:01.000	Apr 30, 2025 @ 17:50:00.000	213.158.93.46	[tb-hosting.com, theatre-saint…	US	R10	Let's Encrypt
Jan 20, 2025 @ 23:19:34.000	Apr 21, 2025 @ 00:19:33.000	217.26.61.226	[equinox-restaurant.ch, hostpoint.ch]	US	R10	Let's Encrypt
Jan 20, 2025 @ 23:19:34.000	Apr 21, 2025 @ 00:19:33.000	217.26.61.226	[equinox-restaurant.ch, hostpoint.ch]	US	R10	Let's Encrypt
Jul 22, 2024 @ 02:00:00.000	Aug 23, 2025 @ 01:59:59.000	217.26.61.226	hostpoint.ch	GB	Sectigo RSA Organization	Sectigo Limited

General Table

15 documents

ingest_timestamp	ip_str	location.country_name	location.city	domains	isp	hostnames	data
Mar 2, 2025 @ 14:09:30.518	217.26.61.226	Switzerland	Zürich	[equinox-restaurant.ch, hostpoint.ch]	Hostpoint AG	[equinox-restaurant.ch, sl2563.web.hostpoint.ch]	HTTP/1.1 200 OK Date: Wed, 26 Feb 2025 19:31:40 GMT Server: Apache…
Mar 2, 2025 @ 14:09:30.473	217.26.61.226	Switzerland	Zürich	hostpoint.ch	Hostpoint AG	sl2563.web.hostpoint.ch	HTTP/1.1 500 Internal Server Error Date: Fri, 14 Feb 2025 05:22:02 GMT Server: Apache…
Mar 2, 2025 @ 14:09:30.429	217.26.61.226	Switzerland	Zürich	hostpoint.ch	Hostpoint AG	sl2563.web.hostpoint.ch	SSH-2.0-OpenSSH_9.7 FreeBSD-20240806 Key type: ssh-rsa Key: …
Mar 2, 2025 @ 14:09:30.366	217.26.61.226	Switzerland	Zürich	hostpoint.ch	SOPRADO GmbH	[sl2563.web.hostpoint.ch, web.hostpoint.ch]	220---------- Welcome to Pure-FTPd [privsep] [TLS] ---------- 220-You are user number 2 of 300 allowed.…
Mar 2, 2025 @ 14:09:29.395	213.158.93.46	Belgium	Antwerpen	[tb-hosting.com, theatre-saint-georges.com, teamblue-ops.net]	Combell NV	[thtres.site.tb-hosting.com, www.theatre-saint-georges.com, theatre-saint-…	HTTP/1.1 200 OK Server: nginx Date: Sat, 01 Mar 2025 01:20:16 GMT…
Mar 2, 2025 @ 14:09:29.275	213.158.93.46	Belgium	Brussels	teamblue-ops.net	Combell NV	213.158.93.46.static.teamblue-ops.net	HTTP/1.1 200 OK Server: nginx Date: Sat, 01 Feb 2025 23:55:30 GMT…
Mar 2, 2025 @ 14:07:30.492	217.26.61.226	Switzerland	Zürich	[equinox-restaurant.ch, hostpoint.ch]	Hostpoint AG	[equinox-restaurant.ch, sl2563.web.hostpoint.ch]	HTTP/1.1 200 OK Date: Wed, 26 Feb 2025 19:31:40 GMT Server: Apache…
Mar 2, 2025 @ 14:07:30.450	217.26.61.226	Switzerland	Zürich	hostpoint.ch	Hostpoint AG	sl2563.web.hostpoint.ch	HTTP/1.1 500 Internal Server Error Date: Fri, 14 Feb 2025 05:22:02 GMT

Figure 3: OSINT Dashboard Overview

Key Dashboard Elements:

Table 4: Key Dashboard Elements

Dashboard Element	**Description**
Total Monitored IPs	Displays a count of unique IP addresses scanned, alongside a timeline chart illustrating the volume of newly ingested events in three-hour increments. This offers a clear snapshot of how many targets are under observation and when they were last updated.
Host Locations (Countries & Cities)	Treemap visualizations break down the distribution of hosts by country and city (e.g., Switzerland, Belgium, Zurich), enabling rapid identification of geographic clusters. This can help uncover unexpected or unauthorized locations.
Identified Ports & Protocols	Color-coded treemaps highlight the top open ports (e.g., 80, 443, 21) and recognized protocols on each monitored host. Security analysts can instantly spot unusual or high-risk ports that may require further investigation.
SSL Certificate Issuers	A pie chart shows the distribution of certificate authorities (CAs) issuing SSL certs for these hosts (e.g., Let's Encrypt, Sectigo). This facilitates auditing of certificate usage and alignment with organizational policies.
SSL Table & General Table	Tabular lists at the bottom display detailed information about each SSL certificate (e.g., issuer, expiry date) as well as a combined view of hostnames, domains, server banners, and last seen timestamps. This tabular approach offers quick drill-downs into specific data points when anomalies are detected.

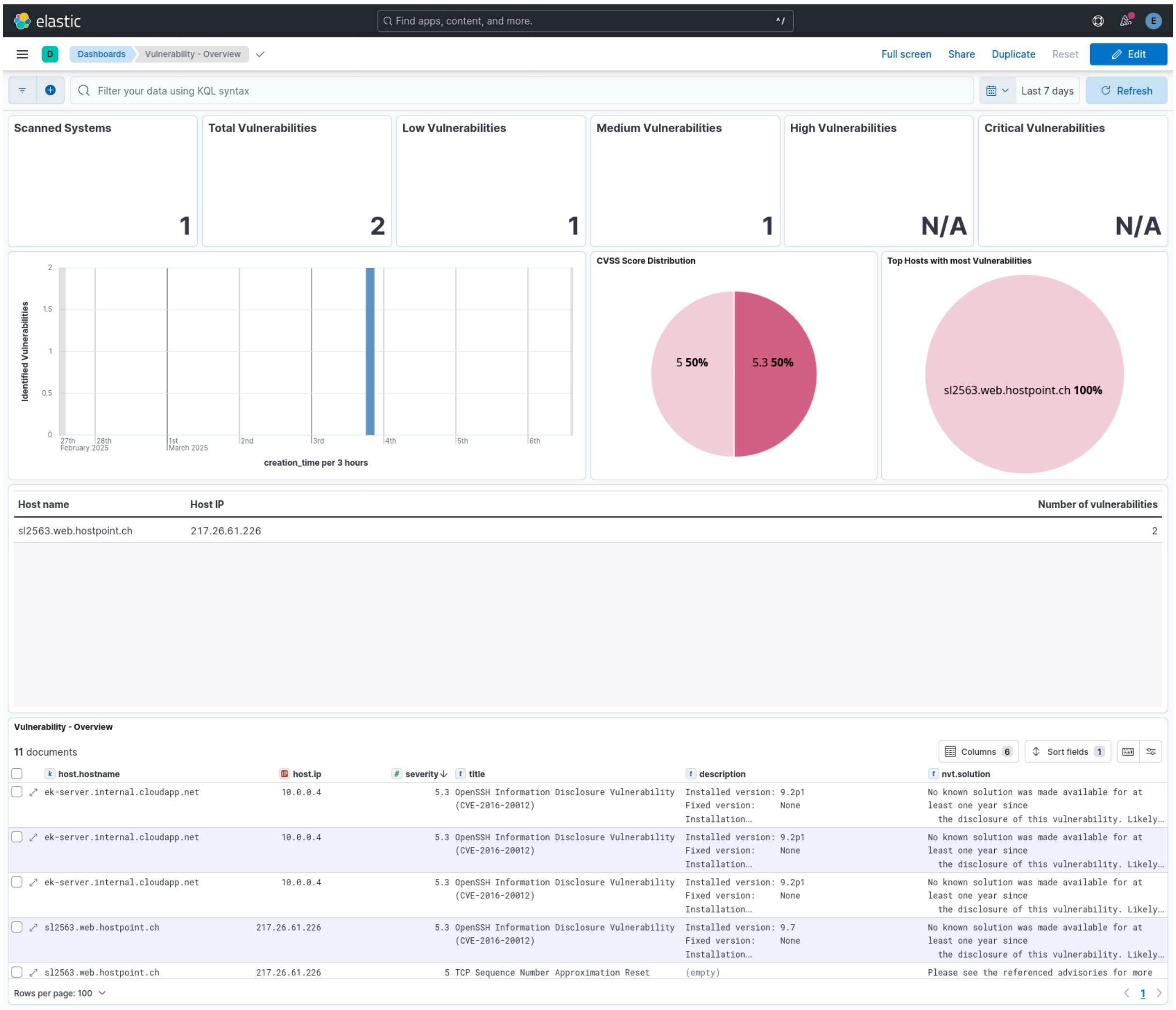
Together, these visualizations provide security teams with clear, actionable insights to quickly respond to anomalies discovered through OSINT data. This dashboard is an example of how our platform consolidates multiple data sources into a unified, easy-to-interpret interface, ultimately reducing the mean time to detection (MTTD) of potential threats.

5.2.2. Vulnerability Management

Our Vulnerability Management Dashboard (see Figure 4) provides an at-a-glance view of all discovered vulnerabilities across all monitored systems. After the Goldoak Vulnerability Worker completes a scan and parses the results, the data is indexed in Elasticsearch and visualized here in near real-time.

Use Cases:

- **Rapid identification of high-risk systems**: Security analysts can immediately identify hosts with critical vulnerabilities, accelerating response.
- **Trend Analysis**: By monitoring spikes in newly discovered issues, teams can identify scanning anomalies or sudden changes in network posture.
- **24/7 automated scans**: Our microservice can schedule scans to identify vulnerabilities at any time.



Scanned Systems	Total Vulnerabilities	Low Vulnerabilities	Medium Vulnerabilities	High Vulnerabilities	Critical Vulnerabilities
1	2	1	1	N/A	N/A

Host name	Host IP	Number of vulnerabilities
sl2563.web.hostpoint.ch	217.26.61.226	2

Vulnerability - Overview

11 documents

host.hostname	host.ip	severity	title	description	nvt.solution
ek-server.internal.cloudapp.net	10.0.0.4	5.3	OpenSSH Information Disclosure Vulnerability (CVE-2016-20012)	Installed version: 9.2p1 Fixed version: None Installation…	No known solution was made available for at least one year since the disclosure of this vulnerability. Likely…
ek-server.internal.cloudapp.net	10.0.0.4	5.3	OpenSSH Information Disclosure Vulnerability (CVE-2016-20012)	Installed version: 9.2p1 Fixed version: None Installation…	No known solution was made available for at least one year since the disclosure of this vulnerability. Likely…
ek-server.internal.cloudapp.net	10.0.0.4	5.3	OpenSSH Information Disclosure Vulnerability (CVE-2016-20012)	Installed version: 9.2p1 Fixed version: None Installation…	No known solution was made available for at least one year since the disclosure of this vulnerability. Likely…
sl2563.web.hostpoint.ch	217.26.61.226	5.3	OpenSSH Information Disclosure Vulnerability (CVE-2016-20012)	Installed version: 9.7 Fixed version: None Installation…	No known solution was made available for at least one year since the disclosure of this vulnerability. Likely…
sl2563.web.hostpoint.ch	217.26.61.226	5	TCP Sequence Number Approximation Reset	(empty)	Please see the referenced advisories for more

Figure 4: OSINT Dashboard Overview

Table 5: Key Dashboard Elements

Dashboard Element	**Description**
High-Level Metrics	Displays the total number of **scanned systems**, as well as the count of **low**, **medium**, **high**, and **critical** vulnerabilities. These metrics update automatically whenever new scan results are ingested.
Timeline Chart	A bar chart indicates when vulnerabilities were first discovered (by creation time), helping security teams identify spikes or trends in newly detected issues.
CVSS Score Distribution	A pie chart visualizes how vulnerabilities are distributed by their CVSS (Common Vulnerability Scoring System) ratings, allowing quick assessment of overall risk severity within the environment.
Top Vulnerable Hosts	Another pie chart highlights which hosts have the highest number of discovered vulnerabilities. This feature enables swift prioritization of remediation efforts on the most affected servers.
Detailed Vulnerability Table	A tabular list at the bottom displays each vulnerability, including: • **Host/IP**: Identifies the specific system affected. • **Severity**: Numeric CVSS score or descriptive rating (e.g., ‘Medium’). • **Title / Description**: Provides vulnerability details (e.g., ‘OpenSSH Information Disclosure’). • **Solution**: Suggests recommended fixes or references for remediation.

Overall, this dashboard empowers security teams to **find and fix** vulnerabilities in the environment and provides a centralized, real-time view of their remediation priorities. In addition, by using Greenbone in this approach, we ensure that we are using a GDPR-compliant vulnerability scanner, allowing us to comply with customer requirements [12].

6. Excursus – Connecting external systems

Beyond our core use of Elastic Stack and our own microservices, we are also able to integrate external EDR/XDR solutions directly into our application. In this example, we will collect logs from endpoints and servers using an EDR/XDR solution “agent” Figure 5. The agent forwards the data to a central server where the data is processed and stored. Using a custom-built logstash pipeline, we are able to ingest the data into our elasticsearch instance. By mapping the resulting events to MITRE ATT&CK techniques, we gain clear visibility into potential threats, enabling proactive threat hunting and streamlined incident response Figure 6.

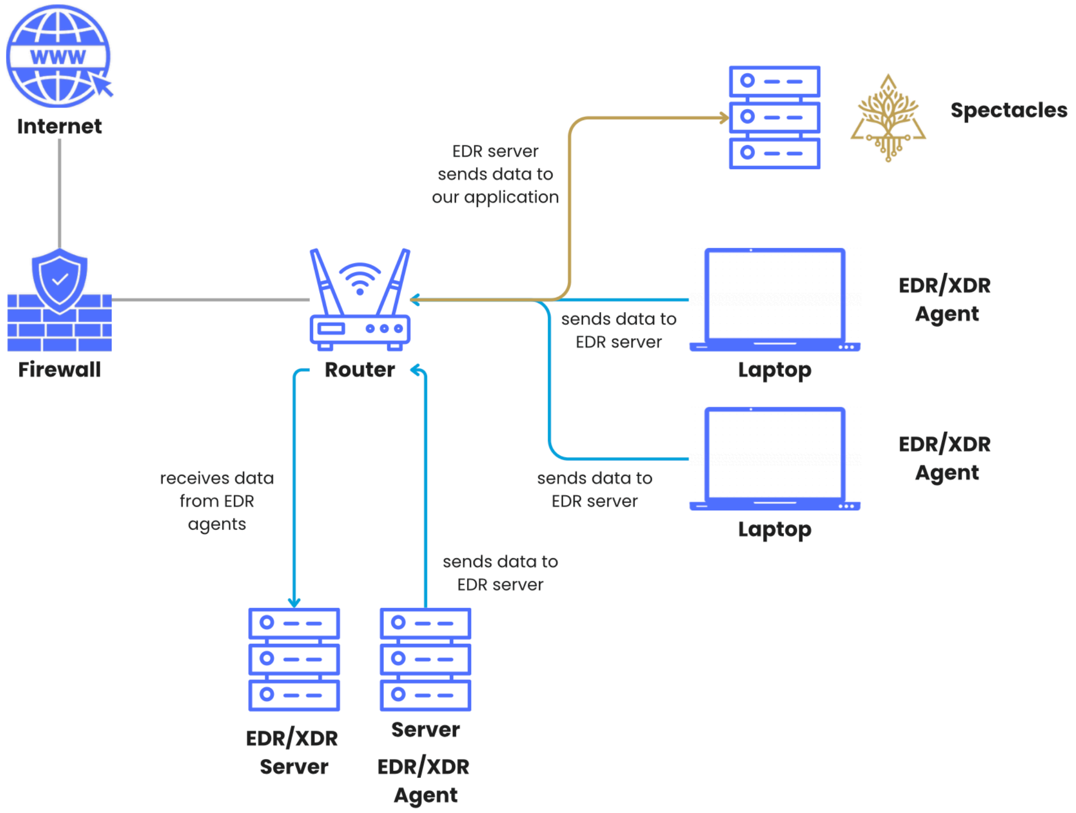
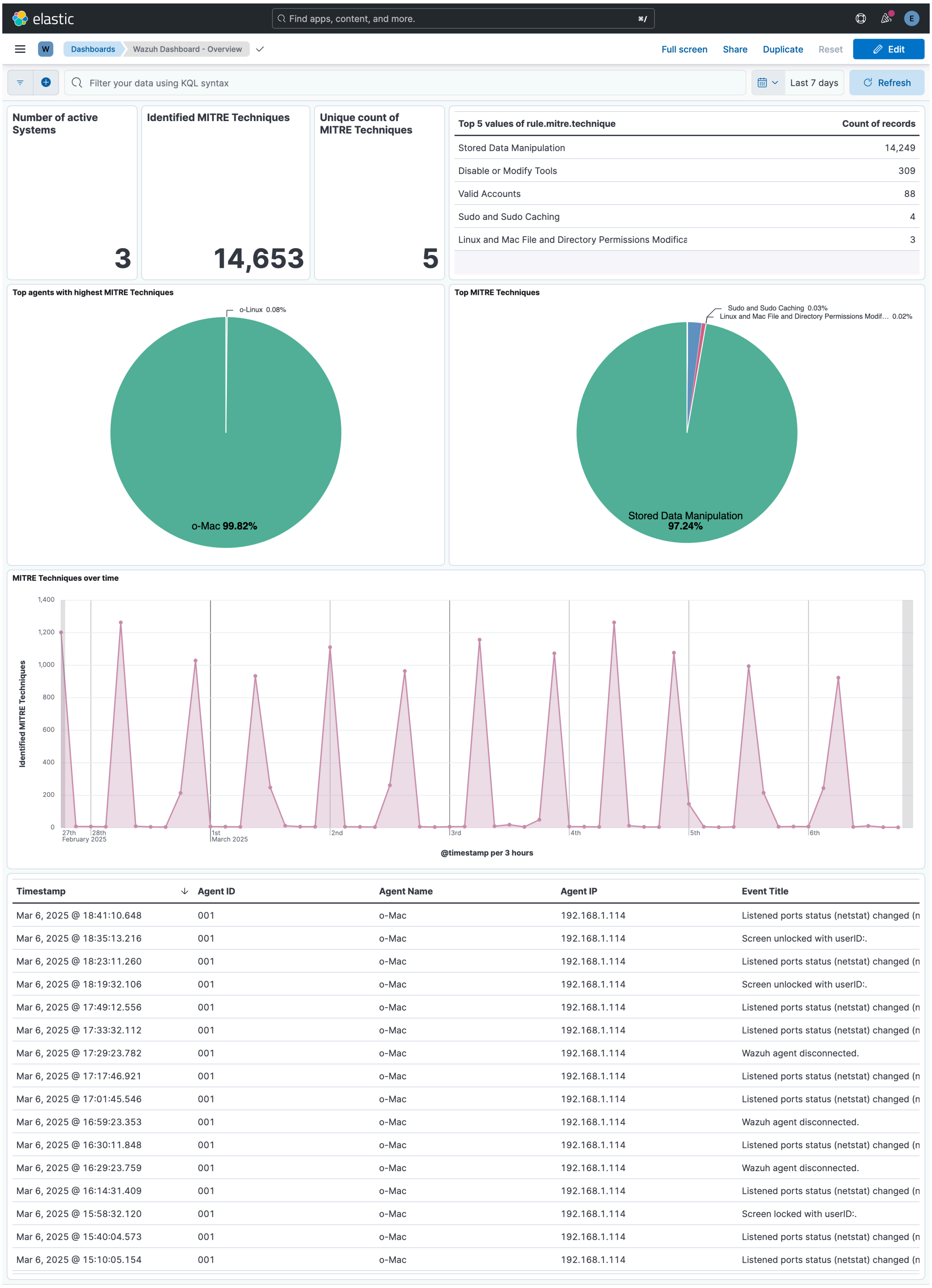


Figure 5: EDR/XDR Network Topology

Figure 6 shows how these events appear in a unified dashboard within Kibana, revealing critical indicators such as stored data manipulation or privilege escalation. This end-to-end approach-Logstash or Python ingestion pipeline, Elasticsearch indexing, and Kibana visualization-underscores the flexibility and scalability of our platform. If further screenshots or videos are needed to clarify this integration, please visit our GitHub repository.


elastic
Find apps, content, and more.
⌘/
W
Dashboards
Wazuh Dashboard - Overview
Full screen
Share
Duplicate
Reset
Edit
Filter your data using KQL syntax
Last 7 days
Refresh

Number of active Systems

3

Identified MITRE Techniques

14,653

Unique count of MITRE Techniques

5

Top 5 values of rule.mitre.technique	Count of records
Stored Data Manipulation	14,249
Disable or Modify Tools	309
Valid Accounts	88
Sudo and Sudo Caching	4
Linux and Mac File and Directory Permissions Modifica	3

Top agents with highest MITRE Techniques

Top MITRE Techniques

Sudo and Sudo Caching 0.03%
Linux and Mac File and Directory Permissions Modif... 0.02%
Stored Data Manipulation 97.24%

MITRE Techniques over time

Timestamp	Agent ID	Agent Name	Agent IP	Event Title
Mar 6, 2025 @ 18:41:10.648	001	o-Mac	192.168.1.114	Listened ports status (netstat) changed (n
Mar 6, 2025 @ 18:35:13.216	001	o-Mac	192.168.1.114	Screen unlocked with userID:.
Mar 6, 2025 @ 18:23:11.260	001	o-Mac	192.168.1.114	Listened ports status (netstat) changed (n
Mar 6, 2025 @ 18:19:32.106	001	o-Mac	192.168.1.114	Screen unlocked with userID:.
Mar 6, 2025 @ 17:49:12.556	001	o-Mac	192.168.1.114	Listened ports status (netstat) changed (n
Mar 6, 2025 @ 17:33:32.112	001	o-Mac	192.168.1.114	Listened ports status (netstat) changed (n
Mar 6, 2025 @ 17:29:23.782	001	o-Mac	192.168.1.114	Wazuh agent disconnected.
Mar 6, 2025 @ 17:17:46.921	001	o-Mac	192.168.1.114	Listened ports status (netstat) changed (n
Mar 6, 2025 @ 17:01:45.546	001	o-Mac	192.168.1.114	Listened ports status (netstat) changed (n
Mar 6, 2025 @ 16:59:23.353	001	o-Mac	192.168.1.114	Wazuh agent disconnected.
Mar 6, 2025 @ 16:30:11.848	001	o-Mac	192.168.1.114	Listened ports status (netstat) changed (n
Mar 6, 2025 @ 16:29:23.759	001	o-Mac	192.168.1.114	Wazuh agent disconnected.
Mar 6, 2025 @ 16:14:31.409	001	o-Mac	192.168.1.114	Listened ports status (netstat) changed (n
Mar 6, 2025 @ 15:58:32.120	001	o-Mac	192.168.1.114	Screen locked with userID:.
Mar 6, 2025 @ 15:40:04.573	001	o-Mac	192.168.1.114	Listened ports status (netstat) changed (n
Mar 6, 2025 @ 15:10:05.154	001	o-Mac	192.168.1.114	Listened ports status (netstat) changed (n

Figure 6: EDR/XDR Network Topology

7. The Goldoak Dashboard - a high-level view of metrics and KPIs

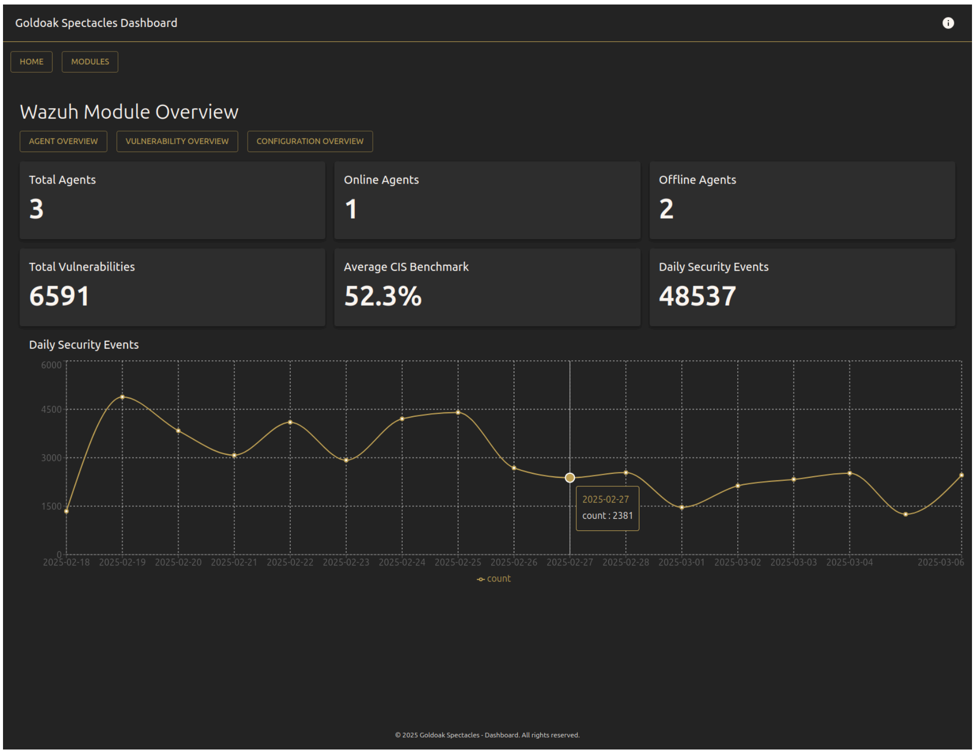
While Kibana provides a deep view of security data and allows analysts to interact directly with logs, the **Goldoak Dashboard** provides a concise, high-level perspective for executives or non-technical stakeholders. This interface highlights key metrics and trends without exposing the granular data fields typically required by security engineers. The result is an at-a-glance solution for monitoring overall security posture and generating periodic reports.

7.1. Purpose and Audience

1. **Senior Management**: Provides decision makers with quick insight into the number of connected systems, ongoing threat events, or open vulnerabilities.
2. **Project Managers/Directors**: Supports monthly or quarterly reporting that shows trends over time and enables teams to measure improvements in their security posture.

7.2. Example View: EDR Events Summary

Below is a sample screenshot (see Figure 7) illustrating how the Goldoak dashboard can display EDR/XDR events at a high level. While technical analysts may rely on Kibana to drill down into the specifics of each event, this dashboard is designed for quick review—enabling top-level metrics such as total events, affected hosts, and summarized threat categories.



Wazuh Module Overview

Total Agents
3

Online Agents
1

Offline Agents
2

Total Vulnerabilities
6591

Average CIS Benchmark
52.3%

Daily Security Events
48537

Daily Security Events

2025-02-27
count : 2381

© 2025 Goldoak Spectacles - Dashboard. All rights reserved.

Figure 7: EDR/XDR Network Topology

From this unified view, executives can quickly identify anomalies, track whether incidents are increasing or decreasing, and assess whether additional resources or escalations may be required. If more detailed forensics are required, a single click takes analysts to the underlying Kibana dashboards for in-depth analysis.

8. Technical Validation & Results

8.1. Testing Environment and Setup

To validate the portability and robustness of our solution, we conducted testing in multiple environments, ranging from cloud platforms to self-hosted and fully on-premises scenarios. Our consistent use of containerization (Docker) and Kubernetes (Minikube for single-node setups) allowed for a unified deployment process, significantly reducing configuration overhead and minimizing inconsistencies between different infrastructures.

8.2. Server Specifications

Table 6: Server Specifications and Roles

Server	**Details**
Server 1 (Main Server)	• **Hardware**: 16 GB RAM, 500 GB Storage, 8 vCPUs • **Role**: Runs Minikube hosting Elasticsearch, Kibana, Goldoak Backend & Frontend, and the OSINT Worker. • **Why**: Provides an all-in-one Kubernetes cluster (single node) for core services.
Server 2 (Dedicated Vulnerability Scanner)	• **Hardware**: 16 GB RAM, 250 GB Storage, 4 vCPUs • **Role**: Hosts the Goldoak-Vulnerability-Worker (Docker) and a Greenbone Vulnerability Scanner (Docker). • **Why**: Isolates vulnerability scanning processes, preventing heavy scan operations from impacting the primary cluster.
Server 3 (External EDR Testing)	• **Hardware**: 8 GB RAM, 250 GB Storage, 4 vCPUs • **Role**: Runs an external EDR solution (Docker) and a dedicated Logstash pipeline (systemd-managed) for data ingestion. • **Why**: Demonstrates how external security tools can forward logs or alerts into our platform's Elasticsearch index using either Logstash or a Python-based microservice.

8.3. Environment Overview

Table 7: Environment Overview

Environment	Description
Azure Cloud	• **OS**: Debian-based VM instances • **Why**: Validated that our platform can scale in a mainstream public cloud environment with minimal setup time.
Self-Hosted Server	• **OS**: Debian-based server • **Why**: Demonstrated ease of configuration in a private data center or personal lab setting, ensuring end-to-end control over hardware and network.
On-Premise Installation	• **OS**: Ubuntu-based server • **Why**: Proved our platform can run seamlessly on local corporate infrastructure (e.g., behind organizational firewalls), maintaining strict compliance and data governance.

8.4. Prerequisites and Automation

Package Updates: All servers were upgraded to the latest operating system packages prior to deployment.

Containerization Tools: Required Docker and Minikube installations are automated using our bash scripts.

Reverse Proxy (nginx): Installed automatically to handle incoming HTTP/HTTPS traffic, with optional Let's Encrypt or custom SSL certificates.

Firewall Configuration: Minimally exposes ports 80 and 443 (for Nginx), plus any custom ports required by external solutions (e.g., EDR). By routing most services through Nginx, the system remains both lean and secure.

8.5. Deployment and Setup Summary

Scripted Install: We provide bash scripts (e.g. `install-app.sh`) that automate Docker and Minikube installation, certificate generation, and firewall rules. This ensures a consistent, out-of-the-box experience across different server environments and requires minimal manual intervention from end users.

Scalability Considerations: While this PoC uses single-node Minikube clusters, the same Kubernetes manifests can be applied to multi-node or cloud-managed Kubernetes, allowing for horizontal scaling in the future. Server resource usage (CPU, RAM) can be tuned as needed - especially for scan-intensive components like Greenbone.

Flexibility and customization: For advanced setups, customers can provide custom SSL certificates or rely on Let's Encrypt for public-facing endpoints. Specialized components (e.g. an external EDR solution) can be integrated via Logstash or a Python microservice, underscoring the platform's adaptability.

In summary, our **multi-environment testing** confirmed that a consistent container-based approach, along with Kubernetes orchestration, enables rapid and repeatable deployments across multiple infrastructures. The system requirements (Docker, Minikube, Nginx) are straightforward, and automated scripts significantly reduce setup complexity, making this solution accessible to smaller organizations with limited IT resources.

8.6. Limitations and Known Issues

While our proof of concept (PoC) has demonstrated significant potential, several limitations remain:

Limited Scale Testing: We have not validated performance under heavy load (e.g., 100-1,000 concurrent systems) due to resource constraints. Although Kubernetes and Docker scale well in theory, more robust stress testing and pilot customers are needed to confirm real-world scalability.

Incomplete certifications and security audits: No formal certifications (e.g., ISO 27001, SOC 2) have been pursued to date. We have conducted internal reviews, but comprehensive penetration testing and third-party security audits are pending.

Small team and accelerated development: Our team size and budget limit the speed and depth of development. QA resources are limited, making it difficult to thoroughly test all potential edge cases or platform integrations.

External dependencies: Certain functionality relies on third-party services (e.g., OSINT APIs, SSL certificate providers). Outages or policy changes to these services could temporarily affect the platform's capabilities.

Limited SLA offerings: As a startup, we are not currently able to offer comprehensive service level agreements (SLAs) for availability or support. Future funding or partnerships may allow for more robust SLAs.

Planned features not yet implemented: Some modules (e.g., advanced threat intelligence, AI-driven analytics) are still in concept or early development. Early adopters may encounter placeholders or limited functionality in these areas.

These limitations reflect both our current stage of development and the inherent constraints of a small, rapidly evolving project. Many of these issues can be addressed with additional funding and partnerships that will allow us to conduct more rigorous testing, pursue relevant certifications, and expand the feature set.

9. Conclusions and Next Steps

This proof of concept (PoC) demonstrates the viability of a containerized, microservices-based cybersecurity platform that seamlessly integrates with various components (e.g., OSINT scanning, vulnerability management, EDR/XDR solutions). By leveraging Kubernetes for orchestration and Elasticsearch for real-time data analytics, we have built a solid technical foundation that can scale to meet the evolving needs of small and medium enterprises (SMEs) as well as larger organizations.

9.1. Key Findings from the PoC

Rapid Deployment & Updates: Our use of Docker and Kubernetes proved effective in quickly packaging, deploying, and updating the platform's various microservices (e.g., OSINT Worker, Vulnerability Worker). Even across different environments (Azure cloud, self-hosted servers, and on-premises), the consistent container-based approach ensured minimal manual configuration.

Real-Time Visibility: Elastic Stack dashboards (Kibana) provided near-instantaneous feedback on critical events, significantly reducing the time required to detect, investigate, and respond to potential threats. This real-time visibility is critical in rapidly evolving attack scenarios.

Extensible Architecture: The microservices framework enables seamless integration with external security solutions, such as EDR/XDR agents or additional vulnerability scanners. It also opens the door to new modules such as threat intelligence or configuration management, allowing each additional service to be developed and deployed independently without impacting existing functionality.

Ease of Adoption: Initial testing across multiple infrastructures (Azure, on-premises Debian/Ubuntu, and self-hosted servers) confirmed that automation scripts and container orchestration keep deployment overhead low. This ensures that organizations with limited IT resources can easily adopt and scale the platform.

9.2. Roadmap for Further Development

Our roadmap focuses on expanding the platform's capabilities while progressively introducing AI-driven automation. Figure 8 below outlines key milestones:

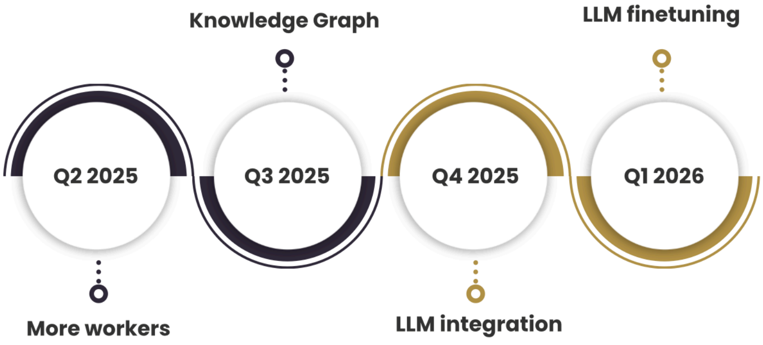


Figure 8: EDR/XDR Network Topology

Table 8: Roadmap for Further Development

Milestone	**Timeline**	**Details**
More Goldoak Workers	Q2 2025	Enhance coverage of security domains, enabling automated data correlation and compliance checks.
Developing a knowledge graph	Q3 2025	Store e.g. MITRE ATT&CK elements, Indicators of Compromise, and real-world cyber attack data (from honey pots or purchased datasets). This high-quality graph serves as the backbone for future AI modules and helps minimize hallucinations by providing a reliable context store.
LLM integration	Q4 2025	Deploy AI-based workers (AI-Vulnerability-Worker, AI-Config-Worker, etc.) capable of analyzing system data and recommending remediation. The LLM will query the knowledge graph via RAG (Retrieval-Augmented Generation) to ensure accurate, context-aware output.
LLM finetuning	Q1 2026	Fine-tune large language models on real (opt-in) customer data, unlocking advanced capabilities such as autonomous vulnerability triage or active response. Early adopter pilots will validate effectiveness and guide further refinements.

9.3. Potential Pilot Deployments / Customer Trials

Collaborative Early Access: We are looking for a select group of pilot customers who are willing to use our platform in a real-world environment. This feedback on functionality, performance and user experience will be invaluable in refining the system.

AI-driven cost savings: By automating repetitive analysis tasks, the AI modules can significantly reduce the cost of security operations and extend the capabilities of existing security teams.

Active Response Integration: Our ultimate vision is to enable AI to take proactive action in real time, such as isolating compromised hosts or updating firewall rules. This will further reduce mean time to containment (MTTC) and alleviate the shortage of skilled security professionals.

In summary, this PoC has laid a solid foundation for a next-generation security platform. Future developments will leverage knowledge graphs and AI-driven workflows, enabling the system to not only observe and report, but also **act intelligently** to thwart emerging cyber threats. This aligns with our mission to deliver accessible, enterprise-level security solutions at a scale and cost that meets the needs of organizations across all industries.

References

[1] Cloud Native Computing Foundation, “Tempestive uses Dapr and Kubernetes to track billions of messages on IoT devices while reducing costs,” 2024, [Online]. Available: https://www.cncf.io/case-studies/tempestive/

[2] Cloud Native Computing Foundation, “Deltatre chose Flux, Helm and K8s to automate and scale,” 2024, [Online]. Available: https://www.cncf.io/case-studies/deltatre/

[3] Cloud Native Computing Foundation, “How Grafana Security is using Dapr to improve vulnerability scanning,” 2024, [Online]. Available: https://www.cncf.io/case-studies/grafana/

[4] Cloud Native Computing Foundation, “CNCF 2023 - Annual Survey,” 2023, [Online]. Available: https://www.cncf.io/reports/cncf-annual-survey-2023/

[5] Elasticsearch B.V., “EY applies search with Elastic to pioneering generative AI experience for finance,” 2024, [Online]. Available: https://www.elastic.co/customers/ey

[6] Elasticsearch B.V., “AHEAD deploys Elastic Security machine learning to decrease triage time, reduce false positives, and automate investigation and response,” 2024, [Online]. Available: https://www.elastic.co/customers/ahead

[7] Elasticsearch B.V., “Global 2000 multinational pharmaceutical corporation accelerates R&D, streamlines pharma industry compliance using Elastic ,” 2025, [Online]. Available: https://www.elastic.co/customers/global2000-multinational-pharmaceutical-corporation

[8] C. Pahl, “Containerisation and the PaaS Cloud,” 2015, [Online]. Available: https://ieeexplore.ieee.org/document/7158965

[9] O. Oyeniran, A. Adewusi, A. Adeleke, L. Akwawa, and C. Azubuko, “Microservices architecture in cloud-native applications: Design patterns and scalability,” *Computer Science & IT Research Journal*, vol. 5, pp. 2107–2124, 2024, doi: 10.51594/csitrj.v5i9.1554.

[10] A., Tiwari and D., Mane, “Application Performance Monitoring Using Log File on ELK Stack,” 2020, [Online]. Available: https://www.irjet.net/archives/V7/i8/IRJET-V7I8726.pdf

[11] A. Voit, A. Stankus, S. Magomedov, and I. Ivanova, “Big Data Processing for Full-Text Search and Visualization with Elasticsearch,” *International Journal of Advanced Computer Science and Applications*, vol. 8, p. , 2017, doi: 10.14569/IJACSA.2017.081211.

[12] M. Feilner, “Schwachstellenmanagement – DSGVO-konform,” 2022, [Online]. Available: https://www.greenbone.net/blog/dsgvo-konformes-schwachstellenmangement-mit-greenbone-cloud-service/