LLM-BASED SEMANTIC THREAT DETECTION IN DOCKER CONTAINER ENVIRONMENTS

Igor Petrović[1,2]*,
[0009-0001-9936-9208]

Mladen Veinović[1],
[0000-0001-6136-1895]

Marko Premović[1],
[0009-0004-4871-9601]

Ivan Tot[2],
[0000-0002-5862-9042]

Boban Stojanović[2]
[0009-0000-5791-4274]

[1]Singidunum University,
Belgrade, Serbia

[2]Ministry of Defence,
Belgrade, Serbia

Abstract:

Simplicity of portability, ease of management, and scalability are significant characteristics of Docker systems, which is why they are widely used in modern business systems, but they also carry certain security risks. One of the effective forms of prevention of potential threats is active monitoring of system operation through logs and the detection of malicious behavior through log messages. However, traditional security systems are focused on predefined rules that are not capable of detecting complex attack patterns. This paper analyzes the effectiveness of applying LLMs in attack detection. The system is implemented in such a way that specially adapted log messages of a custom application in Docker are collected through Rsyslog and then sent to an LLM model, GPT-4.0 mini, for individual and correlation analysis to analyze the maliciousness of log messages. The proposed model proved effective in interpreting both individual log messages and 50 log messages in correlation. For individual analysis, the system achieved an F1 score of 0.92, and for correlation analysis, it achieved an F1 score of 0.935, confirming higher system efficiency when maliciousness of log messages is interpreted within a broader context, which distinguishes it from standard attack detection systems.

Keywords:

Docker Security, LLM, Log Analysis, Malicious Logs, Attack Detection, Brute Force, SQL Injection.

INTRODUCTION

The security of computer systems is becoming an imperative of modern business, given the increasing dependence of business processes. By improving their way of functioning, additional security challenges are imposed, which also leads to the specificity of the security challenges posed to containerized environments. Modern information systems increasingly rely on Docker systems, which enable greater scalability, better management, and easier system migration, but bring new security challenges, especially with advanced attack patterns. Current protection approaches are mainly focused on defining rules on security systems based on which security breaches are interpreted, and to a lesser extent on detailed analysis of log messages which can significantly contribute to the interpretation of them by artificial intelligence.

Correspondence:
Igor Petrović

e-mail:
igor.petrovic@mod.gov.rs

DOI: 10.15308/Sinteza-2026-183-190

Although the primary usage of the LLMs is not the establishment of secure systems, it is possible to assume the contribution of this frequently used technology in this area as well. Within this work, the possibility of applying LLM for semantic detection of threats in containerized solutions was analyzed, on the example of a system for evaluating the malicious nature of log messages generated by Docker containers at an individual level, as well as for evaluating the malicious nature of log messages in correlation with previous log messages.

2. SYSTEM BACKGROUND AND RELATED WORKS

2.1. RELATED WORKS

Previous research into the security of the Docker environment points to frequent vulnerabilities arising from improper privilege controls, unprotected resources, and configuration errors. Automated tools demonstrate the importance of pre-deployment and runtime security measures that can prevent privilege escalation and exploit propagation, which lay the groundwork for layered container protection [1]. In addition to the layered protection of containers, it is necessary to consider the possibility of dynamic log analysis to achieve overall security. The application of large language models (LLMs) in log analysis shows significant potential in anomaly detection and semantic understanding of system behavior. Publications like SmartGuard [2] and RedChronos [3] demonstrate the ability of LLM models to analyze individual logs and contextual links between events, providing explainable diagnostic reports. Overview of LLM-based Event Log Analysis Techniques: A Survey confirms the trend of applying LLM in anomaly detection and RAG methodology, while at the same time points to a gap in real-time application for container logs. Inspired by these works, the implemented system for this research combines the analysis of individual Docker logs with the correlation of the previous 50 events, enabling dynamic and interpretable detection of malicious behavior in real time. In addition, vector databases, such as Qdrant, enable efficient embedding storage and fast real-time lookup, which is of great importance for the integration of LLM models in containerized environments. Qdrant enables scalable and deterministic embedding search, which opens up new possibilities for using the implemented solution, such as forensics. This combines a high-performance database with LLM log analytics and layered Docker security, making our approach unique and practically applicable in real container applications.

2.2. HYPOTHESES AND RESEARCH QUESTIONS

Hypothesis: Using large language models (LLM) for contextual analysis of a sequence of logs significantly improves accuracy and reduces the number of false positives in detecting multi-level cyber threats in containerized environments compared to analyzing individual logs. LLMs are capable of zero-shot classification of unknown and new malicious logs with high accuracy, providing an explanation that is not possible with traditional models.

The research questions focus on establishing and validating the functionality of a centralized system for collecting log messages from Docker containers, implementing and evaluating an LLM-based log analysis system, and assessing system precision and quality using the F1 score.

2.3. DOCKER CONTAINER SECURITY AND CENTRALIZED LOG COLLECTION

Docker containers represent a good alternative form of virtualization that enables isolated running of applications while sharing the core of the operating system, which achieves high portability and speed of operation [4]. Therefore, the establishment of strong security mechanisms and analysis of container logs is important, especially with the possibility of applying advanced models like LLM, which allow the detection of anomalies and early signs of attacks in real time.

To establish the efficiency of a centralized logging system, tools like Rsyslog are often used. Rsyslog is an open-source tool for the transmission and centralized collection of log messages in network environments, which enables reliable delivery, advanced filtering, and integration with databases. Its application in a centralized system for collecting logs enables effective monitoring, analysis, and correlation of security events from different containers and services, which significantly contributes to the detection of threats and the preservation of the integrity of the infrastructure [5]. Centrally collected log messages can be efficiently forwarded to LLM, which, with adequate prompt tuning, would return the security rating of the log message as well as the rating of the log message in correlation with previous log messages in order to detect a potential attack.

3. LLM APPLICATION IN THE ANALYSIS AND EVALUATION OF LOG MESSAGES

Large language models (LLMs) have significant potential in the analysis and evaluation of log messages because they enable semantic understanding of log content, not just their syntactic processing. These models showed significant abilities in understanding, generating, and reasoning over natural language [6]. Current research has begun to focus on their application to security domains, such as malware detection, vulnerability assessment, and log analysis [7], which extends to the ability to find complex dependencies and subtle patterns in textual data, such as log messages generated by Docker containers. Unlike traditional methods that rely on predefined rules, LLMs can recognize patterns of behavior, anomalies, and contextual connections between events [2]. That way, they can assess whether a log message indicates benign or malicious behavior and provide an explanation of the decision.

4. METHODOLOGICAL APPROACH AND SYSTEM DESIGN

To effectively establish a system in which it is possible to check the relevance of LLM in the semantic detection of threats in containerized environments, it is necessary to: implement an application solution in Docker to which it is possible to forward an attack whose log messages could be interpreted by LLM, establish a system of centralized collection of log messages (Rsyslog) and create an adequate prompt tuning model to achieve the highest possible precision of the model in the interpretation of messages, and then placing them in the vector base. Figure 1 shows the flow diagram of the designed system from the log message generated during the operation of the application to the Qdrant database, which can be used for representation and deeper analysis of existing log messages.

4.1. IMPLEMENTATION OF THE TEST APPLICATION

The testing Docker application was created as a web application using the Python programming language with an appropriate frontend that would provide an adequate user interface on which the user would perform various activities on the system that would generate Docker log messages. The application is designed in a simple way to offer the user the possibility of logging into the system, the possibility of uploading files of different extensions, and the possibility of viewing those files. Realization of the possibility, along with adequate recording of activity on the system through log messages, provides further possibility of analysis of the same, in different ways, including sending it to the Rsyslog server and then sending LLM for analysis. The way in which the custom-created application generates log messages, from which it is possible to develop further analysis (based on all the data contained in them: moment of generation, IP addresses, users, access attempts...), is given in Figure 2.

Potential abuses of this application solution can include: brute force attack, SQL injection, as well as malicious file upload in the user part of the application (upload injection). As Docker is increasingly used in modern information systems, the implemented solution is placed in Docker containers, and the Dockerfile that enables this possibility is shown in Listing 1.

The test application can be deployed through the docker-compose.yml file. Listing 2 shows an example of created docker-compose.yml file.

After the successful launch of the application in the containers, every activity on the application leads to the generation of log messages (dataset for this research) that can be further processed and interpreted with the aim of establishing a security system.

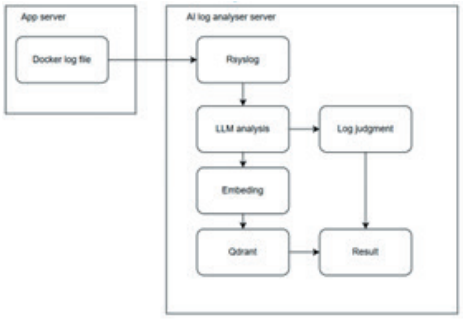

Figure 1. System flowchart

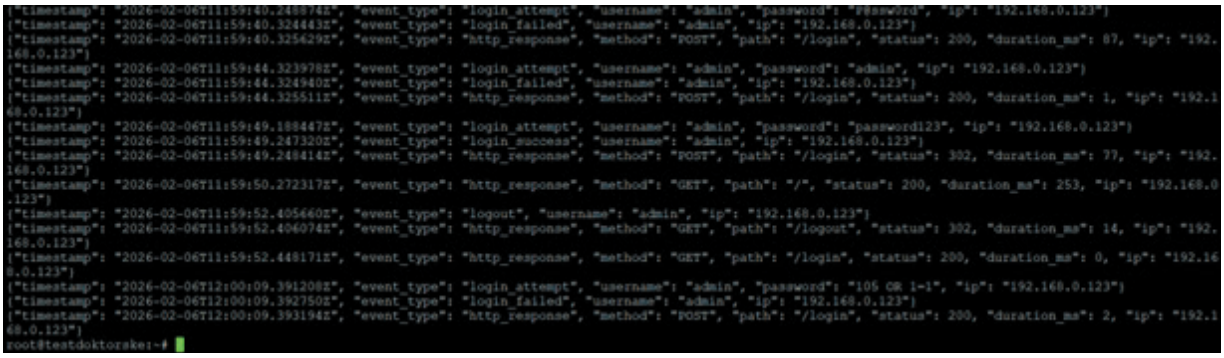

Figure 2. Custom log messages generated by the application in Docker

```
FROM python:3.11-slim
RUN apt-get update && apt-get install -y build-essential libpq-dev && rm -rf /var/lib/apt/lists/*
WORKDIR /app
COPY requirements.txt .
RUN python -m pip install --upgrade pip setuptools wheel \
 && pip install --no-cache-dir -r requirements.txt
COPY . .
RUN mkdir -p /srv/app/documents && chown -R nobody:nogroup /srv/app/documents
ENV FLASK_APP=app.py
CMD ["gunicorn", "--bind", "0.0.0.0:8000", "app:app", "--workers", "1", "--threads", "4"]
```

Listing 1. Dockerfile for created test application

```
version: "3.8"
services:
  db:
    image: postgres:15
    environment:
      POSTGRES_USER: appuser
      POSTGRES_PASSWORD: apppass
      POSTGRES_DB: appdb
    volumes:
      - db_data:/var/lib/postgresql/data
      - ./app/init.sql:/docker-entrypoint-initdb.d/init.sql:ro
    ports:
      - "5432:5432"
  web:
    build: ./app
    environment:
      FLASK_ENV: production
      DATABASE_URL: postgresql://appuser:apppass@db:5432/appdb
      DOCUMENTS_PATH: /srv/app/documents
      SECRET_KEY: SecretP@ssw0rd123
    volumes:
      - /srv/app/documents:/srv/app/documents
    ports:
      - "8000:8000"
    depends_on:
      - db
    logging:
      driver: json-file
      options:
        max-size: "10m"
        max-file: "3"
volumes:
  db_data:
```

Listing 2. Docker-compose.yml file for test application

4.2. IMPLEMENTATION OF RSYSLOG SOLUTION FOR AUTOMATIC LOG COLLECTION

In automated business systems, the centralization of log messages can be extremely important for effective monitoring of activities on the network and systems. Rsyslog, a solution that is used for log centralization in this research, can be presented as a flexible and high-quality logging system that is primarily used on Linux systems, with goal connecting, processing, and forwarding log messages by systems and applications. Since in the test environment, the server on which the application solution is located and the server on which the logic according to LLM is implemented are separated, the implementation of Rsyslog is also required on both systems. The way to set up rsyslog is shown in Listing 3.

By successfully setting up Rsyslog, as a central solution for collecting log messages, on the server side it is possible to access the path of log messages that the server receives from other machines. By establishing the above state, the foundations for the development of a system for dynamic analysis of log messages using LLM have been laid.

5. IMPLEMENTATION OF LLM LOG ANALYSER

In distributed systems that are present in the industry today, and which are characterized by container infrastructures based on Docker, a large number of log messages are generated in a short period of time (every activity on the system generates a log message). Manual analysis of these records is not sustainable, which is why the need for automation is imposed. Therefore, a system has been developed that combines large language models and vector representations of text for automated and semantic analysis of logs in real time. The system allows the classification of logs as malicious, potentially malicious or benign, with the explanation provided by the LLM model.

The LLM log analyzer is implemented in Python and consists of three main components:

- An embedding generation module that converts text logs into numerical vectors that represent the semantic meaning of each record. It supports the OpenAI model, but also has the possibility of further integration with local models in order to increase system security.
- A vector database (Qdrant), which serves for efficient storage of embeddings and metadata about logs. It also enables searching and retrieving semantically similar logs, which can be used in future implementations and system upgrades.

```
sudo apt install rsyslog -y
sudo nano /etc/rsyslog.conf
 - custom implementation for Docker logs
module(load="imudp")
input(type="imudp" port="514")
module(load="imtcp")
input(type="imtcp" port="514")
sudo systemctl enable rsyslog
```

Listing 3. Rsyslog implementation

Table 1. Functional logic of the system

Component	Function	Description
Embedder	Text transformation into vector representation	Uses SentenceTransformer to generate embeddings that represent the semantic content of the log.
QdrantDB	Data storage and search	Stores embeddings and LLM results in a vector database, enabling similarity search and retrieval of previous logs.
LLM Analyzer	Log classification	Using a GPT model, classifies logs into “malicious”, “maybe malicious”, and “benign”, with an explanation.
Contextual analysis	Analysis of the log in relation to previous events	Combines a new log with the last 50 entries from the database, thereby increasing the accuracy and robustness of classification.
Processing module	Real-time monitoring and data ingestion	Continuously reads new lines from the log file and automatically triggers the analysis and database insertion pipeline.

- An analytical module based on LLM, which uses prompt engineering and the GPT-4o-mini model. This system analyzes each log individually and in relation to the previous 50 records (correlating them). Based on the results, a JSON record with classification and explanation is generated.

The functional logic of the implemented system is shown in Table 1.

Listing 4 provides a functional overview of the solution. When a user tries to log in to the system with the credentials username: "105 OR 1=1" and password: "105 OR 1=1", they receive the following system response.

6. METHODOLOGICAL APPROACH AND SYSTEM DESIGN

6.1. SYSTEM PERFORMANCE MEASUREMENT

As the previous chapter presented the technical feasibility of the planned system, it is also necessary to confirm the efficiency of the system. The metric used to evaluate the performance of the model, as well as statistical analysis, is the F1 score, which is often used in assessing the efficiency of smart security systems whose behavior can be assessed by the number of true positives, true negatives, false positives, and false negatives [8] [9]. The F1 score combines two other metrics: precision and recall, to provide a single measure that balances the importance of both. Precision measures how many of the positive predictions of the model were actually correct. It is calculated as the ratio of true positives (TP) to the total number of positive predictions (TP + false positives - FP). Recall or sensitivity measures how many of the total number of true positive cases the model was able to identify. It is calculated as the ratio of true positives (TP) to the total number of true positive cases (TP + false negatives - FN). This metric can also be applied to the implemented system, if logs marked with the rating: "malicious" and "possibly malicious" were grouped into the same group, so that the system is observed in the following way: "Successfully/unsuccessfully interpreted benign log message" and "Successfully/unsuccessfully interpreted malicious/potentially malicious log message". The rating of a system that interpreted malicious log messages as malicious can be marked as a “true positive”, and the rating of a system that interpreted benign log messages as malicious can be marked as a “false positive”. The rating of a system that interpreted malicious log messages as benign can be marked as a “false negative”, and the rating of a system that marked benign log messages as benign can be marked as a “true negative”. To achieve objectivity in the assessment of a system for detecting malicious log messages, the generated log messages must be “intentionally provoked” so that their nature can be precisely determined. This can be achieved with a Python script that automatically manages the application in two ways: it generates 14 SQLi queries, then 14 queries with the user “admin” with a random password (Brute force), and 14 regular queries to the application (successful login).

6.2. RESULTS AND DISCUSSION

After describing how the system works, it is possible to present the results of the system and evaluate it. As part of this research, the system evaluated 42 logs generated by automated activity on a created application driven by a Python script (available in the appendix to the paper), and the way the system evaluates log messages in real-time is as shown in Listing 5.

The complete dataset was sent for analysis by LLM, and the assessment of the maliciousness of the message by LLM. The efficiency ratings of the established system are given in Table 2.

```
„Single judgment: MALICIOUS
|  └─ Reason: The username '105 OR 1=1' suggests an SQL injection attempt, which is a common attack vector used to exploit vulnerabilities in web applications. The failed login event indicates that the attempt was likely made to gain unauthorized access.“
```

Listing 4. Showcase of LLM Log Analyzer functionality (command line output)

```
LOG MESSAGE:
2025-10-31T05:17:46-05:00 testdoktorske docker/srv-web-1[4633]: {"timestamp":
"2025-10-31T10:17:46.369140Z", "event_type": "login_attempt", "username": "admin", "password":
"D<Z9-.KMOvn", "ip": "192.168.0.163"}
SYSTEM OUTPUT:
[NEW LOG]
Log: 2025-10-31T05:17:46-05:00 testdoktorske docker/srv-web-1[4633]: {"timest"admin", "password":
"D<Z9-.KMOvn", "ip": "192.168.0.163"}
├─ Single judgment: MAYBE MALICIOUS
│  └─ Reason: The log entry shows a login attempt with the username 'admin'. It could be an automated
attack or a brute-force attempt. However, withs not definitively malicious.
    "judgment": "maybe malicious",
   "reasoning": "The current log shows a login attempt with a complex password that failed multiple
login attempts. While the password appears strong, the login attempt."
└─ Vector stored in Qdrant (ID: 1761905843)
```

Listing 5. Application results for one log message

Table 2. Results

	Single log				Contextual log			
	TP	FP	TN	FN	TP	FP	TN	FN
Brute force	14	0	0	0	14	0	0	0
SQL injection	14	0	0	0	14	0	0	0
Non-malicious logs	0	5	9	0	8	0	1	5
Sum	**28**	**5**	**9**	**0**	**36**	**0**	**1**	**5**

Based on the data provided, it is possible to calculate the F1 score, according to the following formula:

$$Precision = \frac{TP}{TP + FP} \quad (1)$$

$$Recall = \frac{TP}{TP + FN} \quad (2)$$

$$F1 = \frac{2 \times (Precision \times Recall)}{Precision + Recall} \quad (3)$$

Equation 1. F1 score formula

System rating for analyzing individual Docker log messages: F1=0.92. System rating for analyzing Docker log messages in relation to 50 previous logs: F1=0.935. According to the presented results, the established system showed high accuracy in determining malicious and benign messages, with the system's accuracy being higher in the case of interpreting log messages in correlation with 50 previous messages. For further testing and improvement of the solution, there is a need to increase the number of generated logs, but also the diversity of attacks on the system. Another important security aspect that needs to be emphasized is the sending of log messages for analysis to a publicly available LLM (GPT-4o mini), so a security recommendation for using the solution in real systems would be to redirect log messages for analysis to private LLMs.

7. CONCLUSION

The increasing use of Docker environments imposes the need to establish security solutions that would enable efficient interpretation of the behavior of containerized environments. This paper presents the possibility of combining a centralized system for collecting and storing log messages from Docker containers, then redirecting them to individual analysis and analysis in the context of 50 previous log messages by LLM (model: GPT-4o mini) and finally storing them in a vector database. The established system has proven to be a highly efficient system in predicting the maliciousness of messages, on a sample of generated activities on a custom-prepared application in Docker. Good results (F1 score of 0.92 for individual log messages and F1 score of 0.935 for correlated log messages) confirm the hypothesis set in the paper that: the use of large language models (LLM) for contextual, semantic analysis of a series of logs significantly improves accuracy and reduces the number of false positives in the detection of multi-stage cyber threats in containerized environments compared to the analysis of individual logs. The results also indicate the importance of interpreting log messages in correlation with messages previously generated by the system, to determine the malicious nature of the activity on the

system, which is a great advantage of the implemented system compared to traditional attack detection systems that are based on predefined rules and focus on individual log messages. Further research should be focused on expanding the sample on which the accuracy of the given solution is tested, but also on testing the efficiency of LLMs that can function in closed network environments, which would further increase the overall security of the system.

REFERENCES

[1] A. Durate, N. Antunes, P. Simones & B. Cabral, Detection of Implicit Citations for Sentiment Detection, Universiade de Coimbra, 2017, pp. 18-26.

[2] Zhang, H., Shao, S., Li, S., Zhong, Z., Liu, Y., Qin, Z., & Ren, K., "SmartGuard: Leveraging Large Language Models for Network Attack Detection through Audit Log Analysis and Summarization," *arXiv Preprint: arXiv:2506.16981*, 2025.

[3] C. Li, Z. Zhu & X. Zhang, "RedChronos: A Large Language Model-Based Log Analysis System for Insider Threat Detection in Enterprises," *arXiv Preprint: arXiv:2503.02702v2,* 2025.

[4] S. Upadhya, J. Shetty, R. Rajeshwari & G. Shobha, "A State-of-Art Review of Docker Container Security Issues and Solutions," in *AIJRSTEM 17- 116*, pp. 33-36., 2017.

[5] A. Messina, I. Fontana & G. Giacalone, Log monitoring and analysis with rsyslog and Splunk, Consiglio Nazionale delle Ricerche Istituto di Calcolo e Reti ad Alte Prestazioni, 2015.

[6] T.B. Brown, et al., "Language Models are Few-Shot Learners," *Adv. Neural Inf. Process. Syst.*, vol. 33, p. 1877–1901., 2020.

[7] G. Palma, G. Cecchi, M. Caronna & A. Rizzo, "Large Language Models for Scalable and Explainable Cybersecurity Log Analysis," *J. Cybersecur. Priv*, pp. 5-55, 2025.

[8] B. Bokkena, "Enhancing IT security with LLM-powered predictive threat intelligence," in *Proceedings of the 5th International Conference on Smart Electronics and Communication (ICOSEC)*, Trichy, India, 2024.

[9] M. Entezami, A. Rahimi, S. Moradi & H. Zarei, "A novel framework for detecting anomalies in network security using large language models and deep learning," *Journal of Electrical Systems*, vol. 21, no. 1, p. 294–302., 2025.