

عمر المطيري

FinalThesis Sem2V3.docx

- Final Thesis Submission after Reduce Similarity (Word Format)
- Thesis_(2)_AI901_Dr_Amjad_Qatish_Third Trimester 2025 (Moodle PP)
- Midocean University

Document Details

Submission ID

trn:oid::1:3451871904

Submission Date

Dec 31, 2025, 7:15 PM GMT+4

Download Date

Jan 1, 2026, 2:08 AM GMT+4

File Name

xcod.862750428_011621_3V2meS_sisehTlaniF_عمر_المطيري_02492

File Size

85.6 KB

18 Pages

2,912 Words

17,974 Characters

40% detected as AI

The percentage indicates the combined amount of likely AI-generated text as well as likely AI-generated text that was also likely AI-paraphrased.

Caution: Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.

Detection Groups



17 AI-generated only 40%

Likely AI-generated text from a large-language model.



0 AI-generated text that was AI-paraphrased 0%

Likely AI-generated text that was likely revised using an AI-paraphrase tool or word spinner.

Disclaimer

Our AI writing assessment is designed to help educators identify text that might be prepared by a generative AI tool. Our AI writing assessment may not always be accurate (i.e., our AI models may produce either false positive results or false negative results), so it should not be used as the sole basis for adverse actions against a student. It takes further scrutiny and human judgment in conjunction with an organization's application of its specific academic policies to determine whether any academic misconduct has occurred.

Frequently Asked Questions

How should I interpret Turnitin's AI writing percentage and false positives?

The percentage shown in the AI writing report is the amount of qualifying text within the submission that Turnitin's AI writing detection model determines was either likely AI-generated text from a large-language model or likely AI-generated text that was likely revised using an AI paraphrase tool or word spinner.

False positives (incorrectly flagging human-written text as AI-generated) are a possibility in AI models.

AI detection scores under 20%, which we do not surface in new reports, have a higher likelihood of false positives. To reduce the likelihood of misinterpretation, no score or highlights are attributed and are indicated with an asterisk in the report (*%).

The AI writing percentage should not be the sole basis to determine whether misconduct has occurred. The reviewer/instructor should use the percentage as a means to start a formative conversation with their student and/or use it to examine the submitted assignment in accordance with their school's policies.

What does 'qualifying text' mean?

Our model only processes qualifying text in the form of long-form writing. Long-form writing means individual sentences contained in paragraphs that make up a longer piece of written work, such as an essay, a dissertation, or an article, etc. Qualifying text that has been determined to be likely AI-generated will be highlighted in cyan in the submission, and likely AI-generated and then likely AI-paraphrased will be highlighted purple.

Non-qualifying text, such as bullet points, annotated bibliographies, etc., will not be processed and can create disparity between the submission highlights and the percentage shown.





Cyber Security Threat Hunting Through Log Analysis Using LLM with Agentic Action

Prepared By

Name: OMAR SALEM ALMUTAIRI

IDNo.: [251002148]

Submitted in fulfillment of the requirements for the degree
Master of Science in Artificial Intelligence

At

College of Informatics

Midocean University

UAE

Supervisor Name: Amjad Qatish

College of Informatics, Midocean University, Comoros

[2025- December]

Acknowledgements

I want to express my sincere gratitude to Dr. Amjad Qatish for all of his help, guidance, and inspiration during this project. His expertise in artificial intelligence and cybersecurity has improved the caliber and direction of this thesis. It is also very important to thank the College of Informatics professors for providing the resources needed to finish the project. I would especially like to express my gratitude to the many family members and fellows who helped and advised me during the rigorous testing and development of this work.

Abstract

Advanced dynamic assault vectors with complex new attack surfaces and their agility, which often revolve around the shortcomings of the conventional reactive defense systems. The current AI-based threat hunting system for this project can be created and evaluated in order to anticipate harmful activity in server logs. This system can do more than just match static patterns thanks to the "gpt-oss-120b" Large Language Model (LLM). It can carry out agentic operations, which include autonomous log analysis to generate threat hypotheses and suggest courses of action.

. This research involves a complete dataset of security logs containing simulated Brute Force, Persistence and Command and Control (C2) attacks. The system reached a 100% Detection Rate (Recall) and 0% False Positive Rate (FPR)** on the test dataset for our attacks, and successfully reported breaking complicated attack chains including SSH brute-forcing, hidden cron job and also systemd service unauthorized creation. Additionally, the model was able effectively to produce mitigation scripts including iptables rules and process termination commands. This paper confirms the potential of agentic LLMs as intelligent co-pilots for security analysts to achieve a reduction in the overall "mean time to detect" and "time to contain" in active cyber events.

List of Publications

Table of Contents

Chapter 1: Introduction.....	1
1.1 Problem Definition	1
1.2 Thesis Objectives	1
1.3 Thesis Contribution	2
Chapter 2: Overview.....	2
2.1 Introduction to AI in Cyber Security.....	2
2.2 Traditional Log Analysis Methods	2
2.3 LLM Based Anomaly Detection	3
2.4 Agentic Workflow in Threat Hunting	3
2.5 Conclusion.....	3
Chapter 3: Related Work	4
3.1 Introduction	4
3.2 History And Evolution	4
3.3 Research Gap and Research Focus	5
Chapter 4: Proposed Approach	6
4.1 Data Collection and Description	6
4.2 Data Preprocessing	6
4.3 Prompt Engineering and Agentic Design ...	7
4.4 Model Selection	7
4.5 Evaluation Methodology	7
Chapter 5: Experimental Results and Discussion	8
5.1 Experimental Setup	8
5.2 Cyber Attack Simulation	8
5.3 Model Performance and Metrics	9
5.4 Analysis of Generated Remediation Actions	9
5.5 Discussion.....	10

Chapter 6: Conclusion and Future Work	10
6.1 Conclusion	10
6.2 Future Work	11
References	11

Chapter 1

Introduction

1.1 Problem Definition

Organizations face a significant threat in this age of swiftly developing & evolving cyber threats. Legacy approaches for security depend for most time on signature-based detection and manual log analysis and thus, are fundamentally reactive. The lack of recognition in zero-day or "living off the land" methodologies (using legitimate tools in the system for malicious activity) is one of the most pressing points still affecting the field of attack detection in this regard. In addition, the log data is simply so extensive, that humans experience "analyst overload" as they become fatigued, so they are unable to detect subtler indicators of compromise. An imperative is that these systems be able to be sensitive to what is at hand, not just anomalies, but context and free reasoning as well in assisting with threat hunting.

1.2 Thesis Objectives

This thesis seeks to develop and assess a machine learning-based framework to perform proactive threat hunting applying Large Language Models. The specific objectives are:

- * To create a log analysis pipeline for parsing unstructured security logs (Syslog, Auth, Audit) using **gpt-oss-120b**.
- * to add "agentic action" features that let the model make repair plans, connect events over time, and send alerts.
- * to use cybersecurity metrics like Action Efficacy, Detection Rate, and False Positive Rate (FPR) to see how well the system works.
- * to show how the system can connect actions that have been found to industry standards like the MITRE ATT&CK architecture.

1.3 Thesis Contribution

This study has several important implications for artificial intelligence-focused cybersecurity.

Structure: It suggests a protocol where the LLM acts as a system that can create complex attack chains. For example, it connects a brute force attack with later actions in persistence.

High-Precision Identification: The system achieved a 100% recall of malicious incidents and a 0% false positive rate for the test dataset. It effectively filtered out common system noise, like DHCP requests.

Actionable Intelligence: By generating the necessary mitigation scripts, such as iptables blocks and pkll commands, this solution links detection and response. This is different from current Network Intrusion Detection Systems, which only provide alerts.

Operational Relevance: The study demonstrates the model's application to actual server operations by using a genuine assault scenario (SSH Brute Force to C2 Exfiltration).

Chapter 2 Overview

2.1 Introduction to AI in Cybersecurity

We have transitioned from a rule-based model to an AI-based architecture for prediction and detection of cyber threats. Large Language Models (LLMs) are also poised to revolutionize log analysis since they are trained on huge quantities of unstructured data.

2.2 Traditional Log Analysis Methods

Note that most of the initial log analysis had regular expressions (Regex), with statistical baselines. Although beneficial with known threats, these techniques fail to adapt to the evolving nature of contemporary attacks. As discussed earlier in the proposal, these traditional approaches require some pre-defined rules, which leads to time-consuming and error-prone methods.

2.3 LLM-Based Anomaly Detection

Data available in the literature (e.g., LogLLM, LogRAG, etc.) up until October 2023 shows that LLMs could decode semantic meaning from logs as it were, and that statistical methods, for example, miss outliers and missing statistical information. These models use natural language processing to learn the "narrative" of an attack log, rather than matching keywords.

2.4 Agentic Workflows in Threat Hunting

Multi-agent collaboration frameworks are a key research area. Frameworks like Audit-LLM use decomposing agents to break down complex threats. This thesis elaborates on this by emphasising that we can apply ****agentic action****, which allows the LLM to suggest specific commands that can validate, or remediate, a threat.

2.5 Conclusion

There is evidence in the literature for such a shift from passive analysis to active, agentic threat hunting. This thesis attempts to fill the void in practice by applying a high-parameter open-source framework (`gpt-oss-120b`) to a real intrusion situation.

Chapter 3 Related Work

3.1 Introduction

The accurate and timely prediction of cyber attacks is paramount due to the significant financial and reputational losses associated with security breaches. The development of log analysis for threat hunting is reviewed in this chapter, with a particular emphasis on the transition from conventional pattern-matching algorithms to sophisticated generative AI and multi-agent systems. The analysis identifies gaps in the literature that this thesis fills, especially the lack of reliable, zero-shot systems that can produce Agentic Action scripts for instant confinement.

3.2 Evolution of AI in Log-Based Threat Hunting

3.2.1 Traditional and Deep Learning Methods

Early log analysis techniques mostly used statistical techniques to correlate numerical data from log volumes and packet flows. This led to the development of supervised learning-based systems (like Random Forests and Support Vector Machines) that were trained on pre-labeled datasets (like KDD99). Due to their reliance on static signatures, these architectures typically showed poor performance against zero-day and new attack paths, despite being successful against known threats. Better capabilities for anomaly detection by learning sequential log patterns were made possible by the later development of Deep Learning techniques (e.g., RNNs, Autoencoders like DeepLog). Nevertheless, the volume, speed, and unstructured nature of modern security logs present ongoing challenges for these techniques, leading to scalability issues and elevated false-positive rates.

3.2.2 Harnessing Large Language Models (LLMs)

- The introduction of Transformer-based models like BERT and GPT revolutionized this field by enabling "in-context learning". By going beyond simple keyword or statistical matching, LLMs enable systems to extract context and semantic meaning from unstructured logs. Specialized LLM frameworks aimed toward log analysis have been introduced in recent research:
- LogLLM and LogRAG: These frameworks show how pre-trained LLMs, which frequently incorporate Retrieval-Augmented Generation (RAG) principles, can improve semantic understanding and decrease false positives in log-based anomaly detection.
- LLMeLog: This method uses LLM-based parsing and embedding mechanisms to achieve high detection accuracy even with a small amount of labeled data.
- CTIBench: Efforts have also been made to standardize evaluation, with benchmarks like CTIBench developed to assess LLMs' specific capabilities in Cyber Threat Intelligence (CTI) tasks .

3.2.3 Agentic and Multi-Agent Collaboration Systems

- The current frontier involves transitioning from passive analysis to systems that can reason, plan, and act autonomously. **Multi-agent collaboration frameworks** are a key research area, where specialized LLM agents work together to tackle complex threats. For instance, **Audit-LLM** utilizes decomposing agents and a debate mechanism to enhance log-based insider threat detection and improve the trustworthiness of explanations .

3.3 Research Gap and Thesis Focus

While prior work successfully validated the detection capabilities of LLMs, a significant gap remains in the **end-to-end operationalization** of LLM insights:

1. **Bridging Detection to Response:** Current NIDS and many LLM frameworks prioritize alerting but rarely output immediate, executable containment actions.
2. **Zero-Shot Agentic Efficacy:** There is a lack of rigorous validation showing a large, zero-shot LLM's capacity to perform complex attack reconstruction

and safely generate syntactically correct, multi-layered remediation commands (i.e., true **Agentic Action**).

3. **Contextual Aggregation:** Most evaluation metrics fail to emphasize the LLM's capability to correlate numerous low-fidelity events (e.g., 30+ failed logins) into a single, high-fidelity security incident, which is critical for reducing analyst fatigue.

This thesis directly addresses these gaps by proposing and evaluating an LLM-based framework that centers on **Agentic Action** capabilities, rigorously testing its performance against a full, simulated intrusion lifecycle using the high-parameter gpt-oss-120b model.

Chapter 4 Proposed Approach

4.1 Data Collection and Description

The dataset used for this work is a simulated yet realistic example of the "Linux Server Compromise". The time interval from `Dec 5 10:00:00` to `Dec 5 10:21:00` on the logs. The data is combined from different sources of log:

- * Auth Logs: `sshd` login attempts (successful and failed).
- * System Logs: kernel messages, `systemd`, and `CRON`.
- * Audit Logs: the tool "auditd" entries that record syscalls and process execution ('execve').
- * Command History: its common knowledge that the attacker's commands are recorded and logged in the 'bash' log file.

4.2 Data Preprocessing

The following preprocessing steps were applied to ensure the LLM remained focused on pertinent information:

- * Timestamp Normalization: Unifying all log entries to a standard time format.
- * Noise Filtering: To optimize the context window, low-value logs (like routine kernel heartbeats) were minimized, even though the LLM applies its own filtering.

4.3 Prompt Engineering and Agentic Design

The Prompt Engineering Framework serves as the cornerstone of this approach. The system is managed through the Cyber Defense Analyst persona. This framework is designed for practical applications and outlines the essential design process, detailing how code flows from CDPAs and interacts with the underlying architecture. Based on this structure, the model should:

1. Analyze: Review logs to identify Indicators of Compromise (IoCs).
2. Correlate: Link related logs (e.g., downloading a file from a directory and subsequently executing it).
3. Act: For the given scenario, generate directives focused on mitigation.

4.4 Model Selection

The model for this study will be 'gpt-oss-120b'. This open-source large language model was chosen because it has a huge context window to handle lengthy log files and a large number of parameters that allow sophisticated reasoning.

4.5 Evaluation Methodology

The system is assessed in the manner described below.

- * Detection Accuracy: The proportion of malicious events that are found compared to those that are not.
- * Action Efficacy: The resulting remediation scripts' safety and syntactic accuracy.
- * False Positive Rate: The capacity to disregard harmless background noise, such as dhcpclient queries.

Chapter 5

Experimental Results and Discussion

5.1 Experimental Setup

This required filling the gpt-oss-120b model with a chain of more than 100 mixed log lines.

A whole attack lifecycle was included in these logs, including:

1. First Access: IP 104.28.2.14 SSH Brute Force.
2. Persistence: The creation of malicious systemd services and cron tasks (myservice.service).
3. Exfiltration: Using curl, nc, and DNS tunneling to steal data

5.2 Cyber Attack Attack Simulation :

The attack timeline was faithfully recreated by the model.

Phase 1:

Brute	Force	Detection
-------	-------	-----------

Between 10:00 and 10:01, the model identified more than thirty unsuccessful SSH logins. Log Proof: Dec. 5, 10:00:01 server sshd[1501]: The password for the invalid user admin from 104.28.2.14 failed.

Phase 2: Escalation of Privilege and Persistence The creation of hidden files and other unapproved services was correctly identified by the model. Log Evidence: Dec. 5 10:05:01 server CRON[1600]: (root) CMD (echo="1 bash -c 'bash -i >&/dev/tcp/10.1.1.50/9001 0>&1" >> /var/spool/cron/root) > LLM Analysis: Discovered "Systemd service, cron-backdoors, and hidden binaries... persistence and an increase in privilege.

Phase 3: C2 (Command and Control)

Additionally, the model discovered outgoing connections to the malicious IPs 185.10.10.10 and 13.37.13.37. > LLM Analysis: "Mass network connections... listeners on ports 1337, 4444... firewall disabling."

5.3 Model Performance and Metrics

For this data set, a thorough quantitative analysis of the model response produced ****perfect detection scores****.

Table 1: Quantitative Performance Measures

Metrics	Definition	Result
Detection Rate (Recall)	Percentage of malicious events detected	100%
False Positive Rate (FPR)	% of benign events detected as threats	0%
MITRE Coverage	Mapping of techniques to ATT&CK framework	100%

Benign noise, such as Dec 5 10:20:03 server dhclient[1903]: DHCPREQUEST, was effectively classified by the model with zero false positives (see Table 1).

5.4 Generated Remediation Actions and Analysis

One of the key accomplishments of this thesis is demonstrating "Agentic Action." The model successfully generated a comprehensive remediation script.

LLM Generated Action Plan Snippet:

```
``bash
# Block brute-force source
iptables -A INPUT -s 104.28.2.14 -j DROP

# Kill back-door running processess you know how to kill
pkill -f "bash -i >& /dev/tcp/10.1.1.50/9001"
pkill -f "nc -l -p 1337"

# Remove attacker cron entries
sed -i '/evil_script.sh/d' /var/spool/cron/root
```

Evaluation of Actions:

- **Syntax Correctness:** 100%. The commands iptables, pkill, and sed are each syntactically valid within the Linux operating system.

- 100% completeness with respect for remediation. The application addressed the File System layer (removing cron tasks), Process layer (destroying back doors), and Network layer (IP blocking).

5.5 Discussion

The results show that LLMs are effective at spotting threats and support the idea that they can serve as strong threat hunters.

Noise reduction: The model goes beyond basic keyword matching by recognizing context. For example, it flags `rogue.service` but ignores normal `systemd` launches.

Aggregation: This method helps reduce alert fatigue for analysts by grouping more than thirty SSH logs into one finding, which is called "Brute Force Password Guessing."

What We Can Do: If data exfiltration begins, we can respond right away by generating commands to contain the threat quickly.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

- This thesis presented the gptoss-120b large language model (LLM)-based Cyber Security Threat Hunting system with agentic capabilities. The findings indicate that LLMs can achieve a 100% detection rate for known attack patterns and reduce the false positive rate to zero when analyzing complex server logs. Key outcomes include accurate identification of the three stages of the kill chain—Initial Access, Persistence, and Exfiltration—and the generation of remediation commands that are both operationally safe and syntactically correct. The model distinguished between malicious emulation, such as audit log manipulation, and legitimate system administration activities, such as `dhclient`, by leveraging contextual knowledge.

In summary, this investigation confirmed that LLMs serve as effective components in cyber defense, facilitating integration between detection and response, and demonstrating capabilities beyond those of passive analytic tools.

6.2 Future Work

While the results are promising, there are still a few ways to make them even better:

To enable real-time blocking, connect the model to live data streams, like those from the SIEM API.

Adversarial robustness means testing the model with "poisoned" logs that try to make the LLM ignore threats or create harmful commands.

To cover more of the network, expand the system to analyze both network packets (PCAP) and text logs.

Add a human-in-the-loop feedback loop by letting analysts give input to help the LLM improve its future responses and actions.

References

- 1- S. Akhtar, S. Khan, and S. Parkinson, LLM-based event log analysis techniques: A survey, arXiv preprint arXiv:2502.00677, 2025.
- 2- L. Zhang, T. Jia, M. Jia, and Y. Li, LogRAG: Semi-Supervised Log based Anomaly Detection with Retrieval-Augmented Generation, ResearchGate, 2025.
- 3- A. Fariha et al., Log Anomaly Detection by Leveraging LLM-Based Parsing and Embedding with Attention Mechanism, ResearchGate, 2024.
- 4- C. Song et al., Audit-LLM: Multi-Agent Collaboration for Log-based Insider Threat Detection, arXiv preprint arXiv:2408.08902, 2024.

5- M. Q. Li and B. C. M. Fung, Security Concerns for Large Language Models: A Survey, arXiv preprint arXiv:2501.04437, 2025.

6- M. T. Alam, D. Bhusal, and N. Rastogi, CTIBench: A Benchmark for Evaluating LLMs in Cyber Threat Intelligence, Neural Information Processing Systems, 2024.