# Detecting Security Vulnerabilities in Network Configurations using LLMs

*Jonathan Camilleri*

*Supervisor: Ing. David Debono*

**June - 2025**

**A dissertation submitted to the Institute of Information and Communication Technology in partial fulfilment of the requirements for the degree of BSc (Hons) Multimedia in Software Development**

# Authorship Statement

This dissertation is based on the results of research carried out by myself, is my own composition, and has not been previously presented for any other certified or uncertified qualification.

The research was carried out under the supervision of (name of dissertation tutor –Title, Name and surname)

| | |
|---|---|
| . . . . . . . . . . . . . . . . . . . . | . . . . . . . . . . . . . . . . . . . . |
| Date | Signature |

# Copyright Statement

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

Date Signature

# Acknowledgements

I would like to express my sincere gratitude to my supervisor, Ing. David Debono, for his guidance, support, and valuable insights throughout the development of this dissertation. His expertise and encouragement have been essential in shaping both the research process and the final outcome of this work.

I would also like to extend my heartfelt thanks to my family and friends for their unwavering support, patience, and encouragement during this journey. Their understanding and motivation have been invaluable to me throughout my studies.

# Abstract

This study evaluated the effectiveness of Large Language Models (LLMs) in detecting security misconfigurations in Cisco IOS router configurations, measured against the Center for Internet Security (CIS) Benchmarks. A custom dataset of 80 enterprise-grade configurations was generated, each containing controlled errors or Mistype errors across four protocols: AAA, EIGRP, OSPF and RIP. Three prompt designs of increasing specificity, Broad, Mid-level and Specific, were applied to GPT-4o to assess its compliance-checking capability. All outputs were recorded and manually verified, with accuracy measured using Perfect Prediction (PP) scoring.

Results indicated that prompt design plays a decisive role in model performance. The Broad prompt yielded poor detection accuracy, while the Mid-level prompt showed moderate improvements when referencing CIS guidelines implicitly. The Specific prompt, which included protocol-scoped CIS excerpts, achieved the highest overall accuracy of 61% for misconfigurations and 85% for mistypes. Despite this improvement, the inconsistency of results highlights that current LLMs remain unreliable for production-grade compliance auditing. The contribution of this work lies in demonstrating the impact of prompt specificity on LLM performance, exposing critical gaps in automated compliance detection, and introducing a reusable dataset to support future research in LLM-driven network security analysis.

The study is not without limitations. The dataset covers only four protocol domains, leaving other areas of the CIS Benchmark unexplored. Configurations were synthetic and AI-generated, meaning they do not capture the full variability of real production networks, despite being validated in GNS3. Furthermore, the dataset is limited in size, with 80 configurations sufficient for proof-of-concept testing but not exhaustive as a benchmark. These limitations underscore the need for continued expansion and refinement of datasets and evaluation frameworks in future research.

**Keywords:** Large Language Models, Network Security, Cisco IOS, CIS Benchmarks, Misconfiguration Detection, Prompt Engineering

# Table of Contents

# List of Figures

# List of Abbreviations

| | |
|---|---|
| **LLM** | Large Language Models |
| **CIS** | Center for Internet Security |
| **PP** | Perfect Prediction |

# Chapter 1: Introduction

In this section, **you**, the Student, are expected to state clearly:

(a) the 'problem' or 'question' being researched;

(b) why this topic was chosen;

(c) what motivated the you to choose this topic;

(d) why did you investigate the topic the way you did;

(e) what problem did the you wish to explore;

(f) what is the context for the research?

Percentage amount of words in section: 10 % of Dissertation*

## 1.1 Sub-chapter One

Background goes here. Also you can put in some references [**?**].

Another example of citations [**?**, **?**, **?**].

Here is a sample of table in Table 4.12

Use `\newpage` to force start a new page.

A very quick way to create tables in a point & click environment is to use an online table generator[1]

Use `\enquote` for double-quotes. “This is a sample quote.”

Also can try to refer to this image in **??**. Notice that the `.eps` and `.pdf` format vector graphs are favoured, because:

1. they can be zoomed-in to check the detail.
2. text in such formats are search-able.

Try to insert a math equation as in Equation 1.1. If you wanna try the in-line mathematical, here is a sample $\alpha = \pi \cdot \frac{1}{\Theta}$.

$$e^{ix} = \cos x + i \sin x \tag{1.1}$$

When mention some file formats can use `music.mp3`, `latex.pdf`, etc.

## 1.2 Sub-chapter Two

Contrary to popular belief, Lorem Ipsum is not simply random text. It has roots in a piece of classical Latin literature from 45 BC, making it over 2000 years old. Richard McClintock, a Latin professor at Hampden-Sydney College in Virginia, looked up one of the more obscure Latin words, consectetur, from a Lorem Ipsum passage, and going through the cites of the word in classical literature, discovered the undoubtable source. Lorem Ipsum comes from sections 1.10.32 and 1.10.33 of de Finibus Bonorum et Malorum (The Extremes of Good

This is a todo note which appears in the margin

[1]https://www.tablesgenerator.com

and Evil) by Cicero, written in 45 BC. This book is a treatise on the theory of ethics, very popular during the Renaissance. The first line of Lorem Ipsum, “Lorem ipsum dolor sit amet”.., comes from a line in section 1.10.32.

It is a long established fact that a reader will be distracted by the readable content of a page when looking at its layout. The point of using Lorem Ipsum is that it has a more-or-less normal distribution of letters, as opposed to using ’Content here, content here’, making it look like readable English. Many desktop publishing packages and web page editors now use Lorem Ipsum as their default model text, and a search for ’lorem ipsum’ will uncover many web sites still in their infancy. Various versions have evolved over the years, sometimes by accident, sometimes on purpose (injected humour and the like).

## 1.3 Sub-chapter Three

This todo note appears in line with the text. Todo notes are a great tool to leave comments or notes for self and can then be simply commented out.

Vivamus eget odio tellus. Nam libero augue, eleifend molestie est eu, tempus vehicula magna. Sed dignissim imperdiet urna, non viverra risus blandit in. In lacinia aliquet leo, ut interdum ipsum ornare sed. Praesent ac fringilla justo. Vivamus pharetra non ipsum eget semper. In hac habitasse platea dictumst. Donec neque lectus, ultricies id massa posuere, sodales interdum ante. Sed et nunc vitae urna ornare porttitor nec vitae urna. Curabitur sit amet luctus urna. Nulla porta malesuada rutrum. Pellentesque vitae velit sed odio tincidunt iaculis. Nulla facilisi.

## 1.4 Sub-chapter Four

To create a bulleted list you need to make use of the ”itemize” environment. A LaTeXenvironment is a special section within a page where items as placed. An environment always has a “\begin” and a “\end”. Items you can place in such an environment include:

- enumerate – used to create numbered/letter lists
- itemize – creates a bullet list
- figures – to add images or figures (.eps is the recommended format
- tables – you can create them in tablesgenerator.com then copy the LaTeX code and paste it.
- equations – for those really neat looking mathematical formulae
- this list is not exhaustive...

## 1.5 Sub-chapter Five

The following are some of the most common commands used during your writing. Some characters are reserved and cannot be directly written as in a normal word processor but need to be “escaped”. Then there are other useful commands such as adding a footnote, inserting a new line (to begin a new paragraph after a blank line, adding labels to items for cross-referencing, etc.

- The following are special characters which require a \ before each character so the character is reproduced on the output. The tilde and exponent are even more special.
  - \#
  - \$
  - \% – comment
  - \&
  - _
  - ˜– type \textasciitilde
  - ˆ– type \textasciicircum
- \textbf – bold font
- \textit – italics
- \underline{words to underline}

- \emph – emphasized text. If used within an italicized text, the output is normal font. If used by itself, text in its argument is italicized.

An example highlighting the use of \emph.

- An example of emphasized text with *these words emphasized* within normal font.

- *This is an example where* these three words *are emphasized within italicized font*

# Chapter 2: Literature Review

## 2.1 Introduction

The rise of digital connectivity has made cybersecurity a critical concern for organizations and individuals. With the exponential growth in network complexity, identifying vulnerabilities in network configurations has become a complicated challenge. Traditional tools and methodologies, while effective to a degree, are limited in their ability to adapt to evolving threats. Recent advancements in Artificial Intelligence (AI), particularly Large Language Models (LLMs), have introduced transformative capabilities. These models, trained on vast datasets, demonstrate exceptional proficiency in understanding, analysing, and mitigating complex cybersecurity issues, including vulnerabilities in network configurations. By leveraging their natural language processing capabilities, LLMs offer an innovative approach to interpreting security policies, identifying weaknesses, and suggesting robust defences [2], [3].

LLMs have shown significant promise in vulnerability detection by analysing huge amounts of data, such as system logs, configuration files, and network traffic patterns. For instance, models like GPT-4 and its counterparts are adept at recognizing intricate patterns that may signal potential security risks. Furthermore, the adaptability of these models through fine-tuning allows them to specialize in detecting specific vulnerabilities in diverse network environments. In addition to detection, LLMs play a critical role in generating actionable recommendations to enhance network security. However, their deployment comes with challenges, including the risks associated with misconfigurations or unintended consequences resulting from automated suggestions [2], [4].

The ability of LLMs to process unstructured and structured data alike provides them with a unique edge in network security. By integrating advanced prompt engineering techniques, researchers and practitioners have refined the ways in which LLMs interact with datasets. Such advancements ensure the accuracy and relevance of the insights generated by these models. Despite their potential, however, reliance on LLMs for security assessments raises important questions about their compliance with established standards and benchmarks. This has driven a surge of interest in exploring their capabilities, limitations, and the ethical considerations of deploying such systems in critical network infrastructures [5], [6].

## 2.2 LLMs in Cybersecurity

Artificial Intelligence (AI) is transforming the field of cybersecurity by introducing innovative tools and methodologies to counter increasingly complex cyber threats. It offers creative solutions that not only improve current security frameworks but also open the door for proactive measures against potential breaches. By analysing massive amounts of data at high speeds, AI systems can spot anomalies and potential threats that would be challenging for human analysts to detect in real time.

LLMs have demonstrated significant capabilities in cybersecurity by enhancing network security, automating vulnerability detection, and improving incident response. As mentioned by [6], LLMs can analyse vast amounts of security-related data, including network logs, system configurations, and security advisories, to identify potential threats and weaknesses. They support penetration testing by generating security-related insights and simulating potential attack vectors, enabling organizations to proactively strengthen their defences. Furthermore, as noted in [3], LLMs play a role in user authentication by analysing behavioural patterns to detect anomalies in login attempts, enhancing identity verification processes. Additionally, they contribute to malware detection by scanning files, emails, and web traffic for suspicious activity, helping security teams respond to threats in real-time.

LLMs also enhance incident response workflows by analysing security reports, log files, and threat intelligence to uncover vulnerabilities and detect attack patterns [3]. They can be used to generate automated security policies and assist in compliance monitoring, ensuring that organizations adhere to regulatory standards. By leveraging their advanced natural language processing (NLP) capabilities, LLMs improve real-time threat intelligence gathering, allowing cybersecurity professionals to stay ahead of evolving cyber threats.

## 2.3 Applications of LLMs in Cybersecurity

Cybersecurity analytics depend on processing vast amounts of data to identify patterns and anomalies that indicate potential threats [7]. LLMs analyse unstructured data from multiple sources like online threat intelligence, helping to detect emerging cyber threats and improve decision-making in security operations.

LLMs contribute to vulnerability management by prioritizing vulnerabilities based on their potential impact, freeing up security personnel to concentrate on the most important problem. AI-powered tools can be used to analyse network traffic data or system logs, detecting anomalies that traditional methods might miss. [3], [8].

Incident response benefits significantly from AI-driven automation, which correlates security events, prioritizes alerts, and optimizes response workflows to mitigate cyber threats effectively [3]. Additionally, LLMs streamline forensic analysis by reconstructing attack scenarios, reducing the time and effort required for cybersecurity investigations [8]. This also allowed the facilitation of automated routine cybersecurity tasks, such as log analysis and vulnerability assessments, freeing up human expert to focus on more complex issues [9].

LLMs improve malware detection and classification by analysing code execution patterns, network anomalies, and malicious behaviours [3]. In ransomware mitigation, they help identify attack trends enabling proactive defence measures against evolving threats [10].

AI-driven behavioural analytics enhance user authentication and access management by analysing biometric data, device activity, and login patterns to detect anomalous user behaviour [3]. Machine learning models create behavioural baselines, allowing organizations to identify compromised accounts and insider threats more efficiently [8].

AI also significantly enhances automation in cybersecurity, reducing the manual workload of security professionals. It streamlines repetitive tasks such as monitoring network traffic and analysing alerts, allowing experts to focus on strategic initiatives like threat hunting and incident response [3]. This capability is particularly beneficial for organizations with limited resources, as it enables them to prioritize and mitigate high-risk vulnerabilities more effectively.

## 2.4 Transformer Architecture

The transformative capabilities of LLMs are rooted in its underlying transformer architecture, a deep learning framework designed for sequential data processing. Since the advent of attention, the transformer has become the gold standard for language models. By leveraging multi-head attention and feed-forward networks, transformers process sequences effectively, ensuring scalability and efficiency in handling vast datasets [11]. This architecture allows LLMs to understand the relationship between words and phrases in a given context, allowing it to generate coherent and contextually accurate responses.

A core feature of the transformer architecture is its attention mechanisms. This mechanism calculates similarity scores between queries (representing the current word), keys (representing all words in the sequence), and values (providing additional context). These scores are then used to weigh the importance of each word, enabling model to understand relationships and context within the input [11]. The Transformer architecture, originally designed with an encoder-decoder structure, enables models to capture dependencies across sequences, forming the basis for various language models. BERT which uses an encoder only structure, T5 encoder-decoder structures and GPT decoder only, They all leverage attention mechanisms to generate coherent and contextually relevant responses [12].

## 2.5 Prompt Engineering

### *2.5.1 Prompt Engineering: Concepts and Significance*

Prompt Engineering has emerged as a critical technique in optimizing the performance of Large Language Models (LLMs), enabling them to generate accurate and contextually relevant outputs for a wide range of applications. At its core, it involves the careful formulation and refinement of questions or commands to generate targeted, beneficial responses from AI models. This methodical approach aligns the capabilities of generative AI with human goals and organizational requirements, producing responses that meet specific objectives.

In-Context Learning (ICL) is also an important part of prompt engineering. ICL refers to a process in which a language model is fine-tuned or updated with additional knowledge and information while it is actively deployed and interacting with users or a specific environment. This allows the model to adapt and improve its performance over time based on the context in which it is being used.

An important thing to note is that Prompts should be simple, specific and concise. These qualities ensure that the model provides accurate and relevant responses. Complex queries may introduce ambiguity leading to less precise outputs [6].

### *2.5.2 Previous Work Using Prompt Engineering*

Prompt engineering has been extensively used in the fields of cybersecurity and software vulnerability detection to optimize Large Language Models (LLMs) like ChatGPT. Research has shown that carefully structured prompts significantly enhance vulnerability detection accuracy and efficiency.

The authors in [13] investigated the impact of various prompt designs on vulnerability detection. The researchers demonstrated that structured prompts incorporating auxiliary information such as API call sequences and data flow graphs improve ChatGPT's detection capabilities. Additionally, researchers investigating prompt-enhanced vulnerability detection found that incorporating control flow graphs (CFGs) and data flow graphs (DFGs) into prompts significantly enhances LLMs' ability to identify vulnerabilities. These structural elements help models better understand program semantics, leading to more accurate detection of security weaknesses

In another study, J. Cao et al. analysed the effect of various prompting strategies on automated program repair. Their study introduced an improved prompt template and found that providing clearer contextual details about code functionality, intended fixes, and dataset characteristics led to more accurate detection and repair of errors [14].

A key aspect of prompt engineering in cybersecurity applications is its ability to guide LLMs toward security-specific objectives by structuring tasks that align with real-world security requirements. Research has shown that executing LLMs over structured tasks, such as completing encryption modules or securing hardware components, allows them to better analyse security properties in code. By incorporating prompts with well-defined security intents, LLMs can be effectively steered toward identifying vulnerabilities and improving overall security assessment accuracy [15].

Another advancement in prompt engineering is prompt chaining, where sequential, interconnected prompts are used to guide LLMs through complex security tasks. Studies show that this technique improves penetration testing and threat analysis by making AI-generated responses more structured and actionable [6].

These findings highlight the necessity of precise, structured prompt designs in cybersecurity applications, ensuring that LLMs effectively contribute to threat detection, security compliance, and vulnerability management.

## 2.6 Network Security Standards and Benchmarks

### *2.6.1 Foundations of Network Security Standards*

Network security standards provide robust frameworks to safeguard digital infrastructures against cyber threats, offering essential guidelines for secure network configurations. Among the most prominent standards are NIST SP 800-115, OWASP and CIS benchmarks.

The National Institute of Standards and Technology (NIST) provides guidance for secure information security procedures through its Special Publication 800-115. This document outlines a phased methodology (planning, execution, and post-execution) for conducting thorough information security assessments. It also emphasizes the importance of continuous improvement, ensuring organizations adapt to evolving security challenges while implementing NIST guidelines [4].

The Open Web Application Security Project (OWASP) has established itself as the industry application security standard since 2003. Its OWASP Top 10 serves as a launchpad for addressing the most critical risks, reflecting community data on evolving security threats [4].

Similar to NIST and OWASP, The Centre of Internet Security (CIS) benchmarks offer invaluable guidelines for securing diverse operating systems. These benchmarks encompass a wide array of security recommendations, from essential system configurations to advanced security measures, ensuring systems are resilient against a range of cyber threats [16]. The core principle underlying CIS benchmarks is adherence to best practices, informed by extensive research and expertise. These practices encompass a broad spectrum of security measures, from access control to network configurations, designed to create robust defence layers within IT environments [16].

### *2.6.2 Challenges in Adhering to Standards*

Adhering to network security standards poses challenges for organizations, including resource limitations and maintaining compliance in dynamic IT environments. While frameworks like NIST SP 800-115, OWASP and CIS benchmarks provide robust guidelines, consistent implementation can be complex and resource intensive. Manual implementation remains a significant hurdle. Manual configuration is complex, error-prone, and time consuming, making it difficult to scale as IT environments grow. Additionally, systems that are individually secure may collectively represent vulnerabilities if not properly integrated. Financial and human constraints further hinder compliance with resource limitations often lead to incomplete or delayed adherence to standards. The dynamic nature of cyber threats also necessitates frequent updates to security frameworks. The rapid evolution of cyber threats requires organizations to constantly update their security frameworks, which can be resource-intensive and time-consuming. For example, OWASP's regularly updated Top 10 list requires organizations to adapt their practices to address emerging risks [16].

## 2.7 Dataset Generation

Dataset generation is crucial for training machine learning models in cybersecurity, yet obtaining high-quality labelled data is difficult due to privacy concerns, proprietary restrictions, and evolving threats. Large Language Models (LLMs) offer an efficient solution by generating synthetic datasets that mimic real-world data, reducing dependency on manually curated datasets while ensuring scalability

In cybersecurity, dataset generation is crucial for developing machine learning models for vulnerability detection, malware classification, and network intrusion detection. This approach addresses data scarcity by producing synthetic attack scenarios, filling gaps left by limited real-world datasets. Researchers enhance dataset quality through task decomposition, refining data at different scales , and multi-step generation, structuring complex datasets into smaller chunks and generate a portion at a time [17].

LLMs sometimes hallucinate which can produce significant noise into generated results [17]. Overfitting is another risk when the model is limited to the training data, limiting its practical applicability [12]. Security risks also arise, as adversaries may exploit vulnerabilities in the models design or training data to manipulate its behaviour [3].

To enhance dataset reliability, Zero-shot and chain-of-thought prompting could be used to produce a final "approve/reject" recommendation [18]. Human intervention is also a straightforward strategy where comparing annotations from humans and LLMs guided by the same rules may be able to confirm the codes validity but can lead to considerable labelling costs and can be unrealistic in practical deployment [17].

LLMs contributes well to zero-day exploit detection, enabling models to recognize novel threats [12]. They also simulate cyber-attacks supporting intrusion detection and identifying vulnerabilities in IT infrastructures [3].

### *2.7.1 Dataset Tailoring for Network Configurations*

The reliability of Large Language Models (LLMs) in detecting security vulnerabilities within network configurations depends largely on the quality and relevance of the datasets used for training and evaluation. Network security datasets should accurately reflect real-world scenarios, capturing the intricacies of router configurations, topology constraints, and access control policies. This section explores methods for tailoring datasets that improve the ability of LLMs to synthesize, analyse, and validate network configurations while minimizing security risks.

LLM-generated network configurations often contain syntactic, semantic, and topological errors, which can lead to critical security vulnerabilities. Mondal et al. introduce the concept of Verified Prompt Programming (VPP), in which an LLM-generated configuration is assessed using automated verifiers such as Batfish and topology analysers to ensure correctness [19]. The VPP approach allows iterative refinement of datasets by identifying and correcting configuration errors, thereby reducing reliance on human intervention. This methodology ensures that datasets not only contain valid network configurations but also include common misconfigurations that allow LLMs to learn how to identify and remediate security flaws.

A major issue with existing network configuration datasets is that they often lack coverage of real-world vulnerabilities, making it difficult to assess how LLMs perform when detecting network-based attacks. Abdullin et al. highlight the potential of LLM-driven synthetic dialogue dataset generation, which can be applied to network security research by creating datasets that simulate real-time threat scenarios [20]. These datasets can introduce adversarial configurations, network misconfigurations, and access control flaws, which are crucial for training LLMs to recognize vulnerabilities in network configurations.

A key limitation in network security datasets is their bias toward specific network architectures, which can reduce an LLM's ability to generalize across diverse environments. To overcome this, Netgen employs synthetic dataset generation by manipulating different network topologies, such as the NSFNet and Spanish 5-node networks, allowing the dataset to include a broader range of configurations [21]. Additionally, the Net2Plan tool is used to simulate various network planning scenarios, ensuring that datasets reflect realistic traffic patterns, device constraints, and failure conditions. The process of dataset curation must balance realism with security considerations. According to research on network security and LLMs, datasets containing incorrect or insecure configurations can inadvertently train models to generate faulty or vulnerable outputs. To mitigate this, a combination of synthetic dataset augmentation and real-world dataset validation is necessary. Using ML-assisted dataset labelling, researchers have achieved an 85% feasibility rate in classifying optimal routing configurations, improving the accuracy of models trained for security assessments [21].

## 2.8 Metrics

Evaluating the effectiveness of Large Language Models (LLMs) in detecting security vulnerabilities in network configurations requires well-defined performance metrics. Drawing from prior studies, key assessment areas include accuracy, efficiency, security robustness, and interpretability and usability to ensure LLM-based security solutions are reliable, scalable, and practical for real-world cybersecurity applications. Accuracy and effectiveness play a critical role in measuring how well LLM-generated security assessments align with expected results. One essential metric used in prior evaluations is Perfect Predictions (PP), which classifies an LLM-generated security assessment as correct only if it exactly matches the expected output. This rigorous evaluation standard ensures that predictions are not just approximately correct but entirely accurate, minimizing the risk of false positives or misleading security assessments. This approach has been used in prior research to ensure that vulnerability detection models are held to high standards of precision and correctness.

In addition to accuracy, efficiency and performance determine the practical feasibility of deploying LLM-based security detection tools in real-world environments. Total Execution Time serves as a primary performance metric, measuring how long an LLM takes to analyse a network configuration and produce a vulnerability assessment. Faster response times indicate better scalability and responsiveness, especially for large-scale enterprise networks where real-time security assessments are necessary. Studies evaluating LLM-generated security policies have included execution time as a benchmark to assess computational efficiency and the overall feasibility of integrating such models into cybersecurity workflows [12].

Finally, interpretability and usability are essential to ensuring that LLM-based security detection systems produce outputs that cybersecurity professionals can easily understand and apply. One key method for evaluating usability is Expert Review, where cybersecurity professionals assess LLM-generated policies for effectiveness, clarity, and relevance. Through this process, experts can identify potential gaps, ambiguities, or inconsistencies in the generated security recommendations and suggest refinements to improve their accuracy and compliance with security regulations. This approach ensures that automated vulnerability detection systems remain practical and actionable for security teams, making LLM-driven security solutions more reliable for enterprise cybersecurity use [10]. By integrating these evaluation metrics—Perfect Predictions for accuracy, Execution Time for performance, Formal Property Verification for robustness, and Expert Review for usability, organizations can comprehensively assess the viability of LLM-based security vulnerability detection in network configurations. These metrics ensure that models are not only accurate but also efficient, robust, and interpretable, making them suitable for deployment in real-world cybersecurity applications.

The integration of Large Language Models (LLMs) in detecting security vulnerabilities within network configurations marks a transformative step in the field of cybersecurity. By leveraging their advance natural language processing capabilities, LLMs like ChatGPT and GPT-4 excel in analysing complex datasets, automating tasks, and generating actionable insights, significantly enhancing existing methodologies. Tools like LLM-driven synthetic dataset generation ensure the creation of diverse and realistic data, critical for robust model training and evaluation. Despite all this, challenges such as resource demands, data privacy concerns, and model opacity must be addressed for effective deployment. Enhancing explainability, optimizing resource efficiency, and ensuring compliance with security benchmarks are key steps forward. In summary, LLMs offer innovative solutions for strengthening network security. By refining their applications and aligning them with ethical and technical standards, they hold immense potential for combating evolving cyber threats and ensuring resilient defences.

# Chapter 3: Research Methodology

## 3.1 Introduction

This chapter discusses the methodology used to evaluate the compliance of Large Language Models (LLMs) specifically GPT-4o to the Cisco CIS security benchmarks. Tests were conducted to assess how effectively LLMs can identify security misconfigurations in Cisco configurations and determine adherence to industry standards. The approach taken in this research was primarily quantitative , by measuring the accuracy of the models against a controlled dataset of configurations. In addition, qualitative observations were made regarding the model's explanations and recurring error patterns, providing further insight into their interpretative strengths and weaknesses. The test cases were selected to evaluate the level of specificity required for GPT models to provide accurate responses. Controlled datasets were used to introduce variety while maintaining realism, ensuring that the configurations reflected plausible network scenarios rather than artificial templates. This design allowed the evaluations to replicate a more realistic use case in which an analyst might assess a router configuration under varying conditions. The tests were also highly reproducible, as each could be repeated by simply providing the prompt and configuration and depending on the case, excerpts of the CIS Benchmarks. The inclusion of CIS Benchmarks was critical as they represent an internationally recognised standard for secure configuration practices. By grounding the evaluation in these guidelines, the study ensured that the assessment of GPT models was aligned with authoritative best practices in network security. The following sections detail the dataset design, prompt construction and evaluation process before outlining the methods used to record and analyse results

## 3.2 Hypothesis and Research Pipeline

The following research questions were formulated to support the hypothesis and served as a guide to the methodology 1. What techniques can be used to design a dataset of network configuration with security flaws that can be used for testing purposes? 2. How reliable are LLMs in identifying and flagging security vulnerabilities and misconfigurations for compliance with security standards?

### *3.2.1 Research Pipeline*

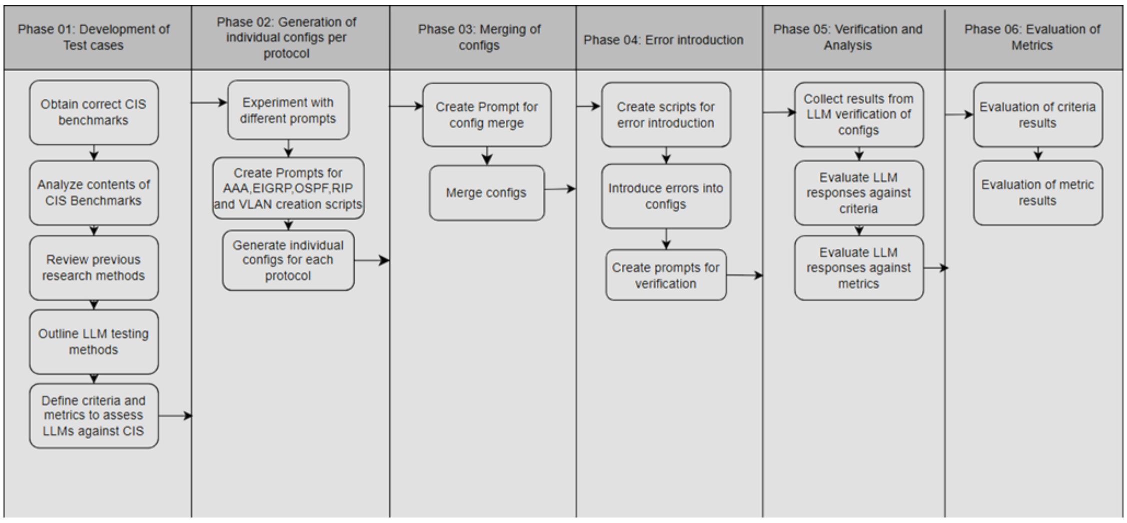

Phase 01: Development of Test cases
Obtain correct CIS benchmarks
Analyze contents of CIS Benchmarks
Review previous research methods
Outline LLM testing methods
Define criteria and metrics to assess LLMs against CIS
Phase 02: Generation of individual configs per protocol
Experiment with different prompts
Create Prompts for AAA,EIGRP,OSPF,RIP and VLAN creation scripts
Generate individual configs for each protocol
Phase 03: Merging of configs
Create Prompt for config merge
Merge configs
Phase 04: Error introduction
Create scripts for error introduction
Introduce errors into configs
Create prompts for verification
Phase 05: Verification and Analysis
Collect results from LLM verification of configs
Evaluate LLM responses against criteria
Evaluate LLM responses against metrics
Phase 06: Evaluation of Metrics
Evaluation of criteria results
Evaluation of metric results

***Figure 3.1:*** *Research Pipeline*

This research followed a structured pipeline designed to ensure reproducibility and alignment with the research objectives. The pipeline consisted of five key stages:

1. Phase 01: Development of Test Cases This phase involved obtaining the CIS Benchmarks, analysing their contents and reviewing related research to inform the design of suitable test cases. From this, the evaluation criteria and performance metrics were defined to provide a consistent basis for assessing the models

2. Phase 02: Generation of individual configs per protocol A series of prompts were developed for generating configurations and for evaluating them under different conditions. Prompts were designed for four different protocols (AAA, EIGRP, OSPF and RIP), for merging multi-protocol configurations, for error introduction and for verification of the configurations.

3. Phase 03: Merging of configs Individual configurations were first generated per protocol before being merged into multi-protocol configurations that more accurately reflected enterprise environments. Errors were then systematically introduced, aligned with CIS requirements for misconfigurations and supplemented with Mistype errors to simulate real-world scenarios.

4. Phase 04: Error introduction Model responses were collected for each test case and assessed against the defined criteria and metrics.

5. Phase 05: Verification and Analysis The results were evaluated at two levels: a. Criteria-based evaluation was performed by determining whether CIS-aligned misconfigurations were correctly identified b. Metric-based evaluation was done by quantifying accuracy across test cases and protocols

6. Phase 06: Evaluation of Metrics Results were evaluated and recorded ac-

cordingly then calculated in PP Score for final scoring.

## 3.3 Dataset Design and Preparation

A custom dataset of Cisco IOS 15 configurations was created for the purpose of evaluating the ability of large language models to detect security misconfigurations against the CIS Benchmarks. The dataset forms the foundation of this research, providing a structured yet realistic set of test inputs that enabled repeatable and controlled evaluation of the models. Since no openly available configuration study.

### *3.3.1 Protocol Coverage and Scope*

The dataset focused on four protocol domains explicitly covered in the CIS Benchmarks and widely deployed in enterprise environments: AAA, EIGRP, OSPF and RIP. For each protocol 20 synthetic configurations were created, giving a balanced distribution across protocol types. The decision to focus on these four areas was taken since they represent high-impact configuration domains where errors or omissions can have serious security consequences, while also being sufficiently well documented within the CIS Benchmarks to support structured evaluation.

### *3.3.2 Dataset Generation*

The dataset was generated using OpenAI's ChatGPT API with carefully engineered prompts. These prompts were designed to mimic the work of a network architect constructing realistic Cisco IOS 15 configurations for Cisco C7200 routers. For example, the instructions required the generation of router configurations with randomised OSPF area IDs, unique hostnames and unique subnets drawn from the 192.168.*.*/24 address space. Interfaces were configured consistently with the advertised networks, and authentication commands were randomly generated to reflect operational diversity. Each generation produced 10 distinct router configurations with the resulting dataset consisting of 80 unique configurations, each approximately 150 lines in length The use of prompts ensured that configurations were not repetitive or based on rigid templates but instead reflected the complexity of production-grade enterprise environments. Features such as VLANs, multiple routing protocols and varied authentication methods were incorporated, ensuring that the dataset simulated the types of configurations typically encountered in corporate networks. The configurations were subsequently tested in GNS3 on C7200 router images to confirm that they were syntactically correct and operationally valid.

### *3.3.3 Prompt Structure*

The prompts used to generate configurations were carefully engineered and divided into a number of parts. The first part was the introduction, which established the role of the model as a network architect and explicitly instructed it to rely only on the attached Cisco manuals for accuracy. This framing ensured that the generated configurations were grounded in authoritative sources rather than the models general training. Figure 3.2 shows the introduction used

You are a network architect.
Use ONLY the following context from the provided PDF manuals—do not rely on any other knowledge.
Output them one after the other.
Each block MUST start with for example ”hostname R1” ”hostname R2” etc.
For each router, assign it a unique hostname (e.g. R1, R2...).
For each router, assign it **unique** IP addresses for each interface.
Configure AAA

***Figure 3.2:*** *Prompt Introduction*

To ensure that the generated configurations reflected valid IOS 15 syntax and realistic command structures, the official Cisco documentation for each protocol was used as the authoritative reference. The manuals were first parsed into smaller chunks using a recursive text-splitting algorithm. These chunks were then embedded using OpenAI’s text-embedding-ada-002 model and indexed in a FAISS vector store. During prompt construction, the retrieval system provided the most relevant documentation snippets, which were injected into the context block of each generation prompt. By embedding and retrieving from the Cisco IOS 15 manuals, the prompts were grounded in verified vendor syntax’s, reducing hallucinated commands and ensuring that the outputs were both syntactically correct and aligned with Cisco’s best practices.

The second part defined the mandatory configuration elements, namely the core AAA commands that had to be introduced on every router. These were essential security features such as “aaa new-model”, authentication lists, accounting rules and encryption settings, all aligned with CIS benchmark requirements. The core commands list is a set of mandatory commands that was required on the AAA configuration to ensure baseline compliance with CIS-recommended security practices. For AAA, these included the commands shown in Figure 2. This approach was applied only to the AAA configurations, as for the other protocols the required commands were embedded directly in the prompt instructions. In the case of AAA, including them in the prompt cause GPT-4o to become confused, often leading it to hallucinate commands or omit required lines entirely. To avoid this issue, the commands for AAA were handled separately. The full list of core commands can be found in Appendix B

```
"aaa new-model",
"aaa authentication login LOGIN-LIST group tacacs+ local",
"aaa authentication enable default group tacacs+ enable",
"aaa accounting connection CONN-ACC start-stop group tacacs+",
"aaa authorization exec EXEC-LIST group tacacs+",
"aaa accounting exec EXEC-ACC start-stop group tacacs+",
"aaa accounting network NET-ACC start-stop group tacacs+",
"aaa accounting system default start-stop group tacacs+",
"aaa accounting vrrs default start-stop group tacacs+",
"aaa accounting delay-start",
"Assign the TACACS+ server host IP to any address in the 10.0.0.0/24
subnet that has an interface associated with it.",
"Configure an 11-character randomly generated alphanumeric secret for
the TACACS+ server.",
"service password-encryption",
"Apply exec-timeout 10 0 to line vty 0 4",
"Apply transport input ssh to line vty 0 4",
"Apply access-class 10 in to line vty 0 4",
"Apply access-list 10 permit with an ip range 10.0.0.0 0.0.0.255"
```

***Figure 3.3:*** *Core commands*

Embedding these rules directly in the prompt guaranteed their inclusion, forming a reliable baseline for error injection later. To introduce variability the third part of the prompt provided an extension block which contained a larger pool of optional but relevant commands. During generation, a subset of these extensions was sampled randomly, ensuring that while all configurations adhered to security best practices, no two files were identical. The extended block (`ext_block`) is a larger pool of optional protocol specific commands designed to increase diversity. At runtime, nine of these domain specific commands were sampled randomly using python. An example of the domain specific commands is shown in Figure 3. This ensured that while all configurations followed a standard structure, no two were identical, better reflecting the variability of real-world deployments. The rest of the `ext_block` commands for other protocols can be found in Appendix C

”Configure a login banner with the text Ẅelcome to my C7200 UNAUTHORIZED ACCESS IS PROHIBITED. This device is monitored.̈.”,
”Configure a failed-login banner with the text N̈ice try.̈.”,
”Set the login timeout to any value between 1 and 10000 seconds.”,
”Configure the RADIUS source interface to use an existing router interface.”,
”AAA accounting for connection events with the method list CONN-ACC.”,
”Configure AAA accounting for EXEC shell sessions with EXEC-ACC.”,
”Configure AAA accounting for network services with NET-ACC.”,
”Configure AAA accounting for system events with the default method list.”,
”Under ‘line con 0‘, configure ‘login authentication LOGIN-LIST‘.”,
”Under ‘line aux 0‘ configure ‘login authentication LOGIN-LIST”,
”Under ‘line vty 0 4‘ configure ‘login authentication EXEC-LIST”,
”Apply aaa authorization config-commands”,
”aaa authorization reverse-access default group tacacs+”,
”aaa authorization commands 15 default group tacacs+ if-authenticated”,
”aaa authentication login LOCAL-CASE local-case”

***Figure 3.4:*** *Extension block*

The fourth part of the prompt inserted context snippets taken directly from Cisco IOS manuals, which helped the model reproduce valid syntax and realistic command sequences. Following this, the IOS template was provided as a skeleton configuration as shown in Figure 4 and Appendix A, giving the model a consistent starting point while leaving sections to be filled in dynamically.

```
!
!
!
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname R1
!
ip cef
no ip domain-lookup
no ip icmp rate-limit unreachable
ip tcp synwait 5
no cdp log mismatch duplex
!
line con 0
exec-timeout 0 0
logging synchronous
privilege level 15
no login
line aux 0
exec-timeout 0 0
logging synchronous
privilege level 15
no login
!
!
end
```

***Figure 3.5:*** *IOS template*

Figure 4: IOS Template Finally, the instruction block specified the formatting and validation rules, unique hostnames, unique subnets, interface activation and strict adherence to both the core and extension commands. This multi-part structure ensured that the generated outputs were syntactically correct, benchmark-compliant and operationally realistic while still maintaining diversity across the dataset.

```
Fill in or extend the attached C7200 Router IOS template above so that
it configures:
- Available Interfaces are: interface FastEthernet0/0, interface Ethernet1/0,
interface Ethernet1/1, interface Ethernet1/2, interface Ethernet1/3, interface
Serial2/0, interface Serial2/1, interface Serial2/2, interface Serial2/3, inter-
face Serial2/4, interface Serial2/5, interface Serial2/6, interface Serial2/7
with the exception of subinterfaces like interface FastEthernet0/0.10
- DO NOT OMIT any command listed above—both CORE and OP-
TIONAL.
- If the context lacks any required command, leave that section blank
and write UNKNOWN.
- List no shutdown on every interface
- Assign each interface a unique /24 subnet in the 10.0.0.X/24 range
- **Output only** the final, completed CLI configuration (no explana-
tions), **Create 10 router configurations**
```

***Figure 3.6:*** *Instruction Block*

The multi-part prompt structure was reused for every protocol domain, with each version tailored to include the relevant benchmark-mandated commands and protocol-specific extension blocks. In this way, the dataset was generated consistently across 80 configurations while still capturing the variability and complexity of real-world enterprise deployments. After protocol specific configurations were generated, they were combined into composite multi-protocol files using a dedicated merging prompt. This ensured that final configurations reflected the complexity of enterprise deployments, where routers rarely operate with only one protocol enabled. An excerpt of the merging prompt is shown in Figure 6

”You are a Cisco IOS network configuration assistant. ”
”You will be provided with multiple protocol-specific configurations (OSPF, EIGRP, RIP, AAA) and a list of interface definitions (including VLAN subinterfaces). ”
”Your task is to merge all of them into a single IOS configuration file, combining all interface blocks when multiple protocols affect the same interface. ”

” IMPORTANT RULES:”
”- If all physical interfaces are used by OSPF, use the defined VLAN subinterfaces (e.g., FastEthernet0/0.x) for EIGRP.”
”- RIP and EIGRP can overlap with other protocols, but should minimize this.”
”- Keep EIGRP keychains under the chosen EIGRP interfaces” ”- Keep ALL authentication under the appropriate interface if it is there already ”
”- OSPF interfaces must be used exactly as defined and must not be reused for EIGRP.”
”- AAA server IPs must use subnets where the router has the .1 IP and the server has .2.”

” Additional Instructions:
” ”- All interface blocks must be merged together (no duplicates).”
”- Subinterfaces (e.g., FastEthernet0/0.100) behave like normal interfaces.”
”- Order the final config as follows: OSPF $\rightarrow$ EIGRP $\rightarrow$ RIP $\rightarrow$ AAA.”
”- If any protocol config is missing, include a placeholder: ’¡protocol¿ section: [UNKNOWN]’.”
”- Output ONLY the final merged configuration text, no explanation.”

***Figure 3.7:*** *Merging Prompt*

This prompt ensured that all generated configurations were integrated into a single coherent file while preserving protocol-specific constraints. Importantly, it prevented unrealistic overlaps, such as reusing OSPF interfaces for EIGRP, while allowing controlled overlaps between RIP and EIGRP to reflect practical scenarios. VLAN sub-interfaces were introduced dynamically when all physical interfaces were exhausted, maintaining a consistent pool of usable interfaces without manual intervention. The prompt also enforced ordering of sections ( OSPF -

EIGRP - RIP - AAA) to maintain uniformity across all configurations. This structured merging step was critical in transforming single-protocol files into enterprise-grade composite configurations of approximately 150 lines, representing realistic scenarios where multiple routing protocols and AAA services coexist.

### *3.3.4 Error Design*

For each type of protocol, the following configurations containing errors were created:

- Five configurations with a single misconfiguration
- Five with two misconfigurations
- Five with three misconfigurations, and
- Five containing Mistype errors or syntatic errors.

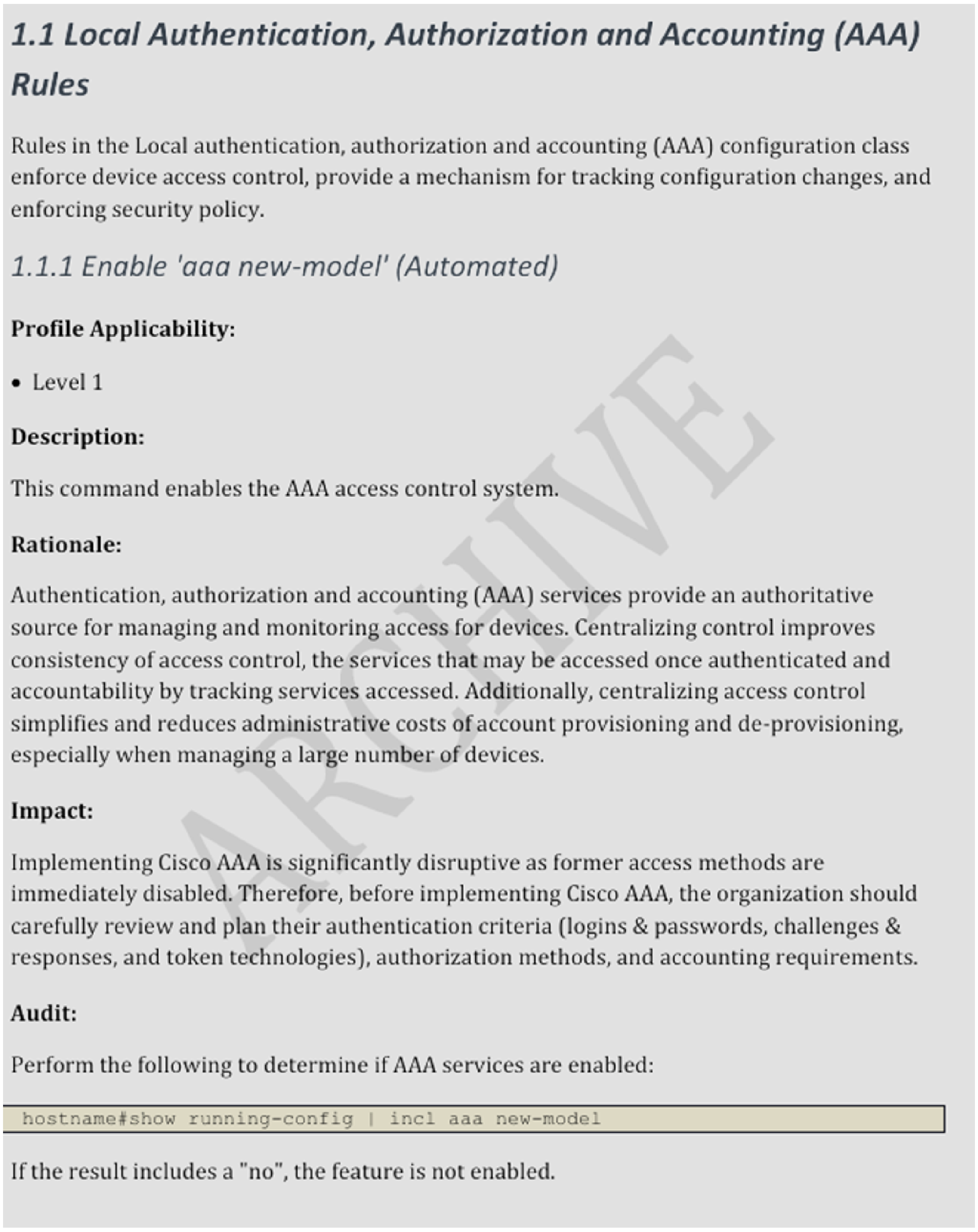
The misconfigurations were created programmatically using Python scripts that systematically removed commands listed in the benchmarks. Each protocol was associated with a curated list of rules expressed as regular expressions as listed in Appendices B and C. This ensured that every removed command directly corresponded to a CIS benchmark requirement, maintaining fidelity to recognised security standards. Figure 7 shows an excerpt of the regular expressions used to introduce errors for the AAA protocol, The protocol lists are referenced in Appendix D

```
r'^aaa new-model',
r'^aaa authentication login LOGIN-LIST group tacacs\+
   local$',
r'^aaa authentication enable .*',
r'^aaa authentication dot1x .*',
r'^aaa authentication ppp .*',
r'^aaa authentication arap .*',
r'^aaa authentication attempts max-fail .*',
r'^aaa authorization exec .*',
r'^aaa authorization config-commands.*',
r'^aaa authorization network .*',
r'^aaa authorization reverse-access .*',
r'^aaa accounting exec EXEC-ACC start-stop group tacacs\+$',
r'^aaa accounting commands 15 .*',
```

***Figure 3.8:*** *Commands for error introduction*

These deletions correspond to essential CIS requirements for enabling AAA services, protecting administrative access and securing SNMP. An example of this would be for the first command "aaa new-model". In the CIS Benchmarks it specifically states "Globally enable authentication, authorization and accounting (AAA) using the new-model command" [1], as shown in Figure 8

By varying the number of removed commands, configurations with different levels of non-compliance were produced, ranging from minor omissions to critical security weaknesses. By contrast, the Mistype errors were not aligned with CIS Benchmarks but were deliberately introduced to test the model's robustness in identifying realistic human mistakes, such as misspelled commands (interfce instead of interface). The script was provided a list of text adjustments. A complete list can be found in Appendix E. This dual approach ensured that the dataset tested both benchmark-related compliance failures and more practical day-to-day issues faced by network engineers.



## 1.1 Local Authentication, Authorization and Accounting (AAA) Rules

Rules in the Local authentication, authorization and accounting (AAA) configuration class enforce device access control, provide a mechanism for tracking configuration changes, and enforcing security policy.

### 1.1.1 Enable 'aaa new-model' (Automated)

**Profile Applicability:**

- Level 1

**Description:**

This command enables the AAA access control system.

**Rationale:**

Authentication, authorization and accounting (AAA) services provide an authoritative source for managing and monitoring access for devices. Centralizing control improves consistency of access control, the services that may be accessed once authenticated and accountability by tracking services accessed. Additionally, centralizing access control simplifies and reduces administrative costs of account provisioning and de-provisioning, especially when managing a large number of devices.

**Impact:**

Implementing Cisco AAA is significantly disruptive as former access methods are immediately disabled. Therefore, before implementing Cisco AAA, the organization should carefully review and plan their authentication criteria (logins & passwords, challenges & responses, and token technologies), authorization methods, and accounting requirements.

**Audit:**

Perform the following to determine if AAA services are enabled:

```
hostname#show running-config | incl aaa new-model
```

If the result includes a "no", the feature is not enabled.

***Figure 3.9:*** *CIS AAA Example [1]*

### *3.3.5 Validation Against CIS Benchmarks*

To ensure that the misconfigurations represented genuine violations, the removed or altered commands were cross-checked against the official CIS Cisco IOS Benchmark for IOS 15. This validation process ensured that the dataset-maintained fidelity to recognised industry standards and that any error introduced would be considered non-compliant under CIS rules. In this way, the dataset served not only as a testbed for GPT models, but also as a controlled approximation of the compliance-checking process used in professional network security audits.

### *3.3.6 Dataset Realism*

To replicate real-world conditions, the configurations were not isolated per protocol but instead merged into composite files containing multiple features, for instance, an OSPF-focused test configuration also included AAA and EIGRP sections, reflecting the reality that enterprise routers rarely operate with a single protocol enabled. This approach ensured that the dataset challenged the models to parse through complex, multi-protocol configurations in order to identify the relevant issues.

## 3.4 Prompt Construction

Prompt construction was central to this study, as it determined how GPT-4o were guided in their evaluations of Cisco IOS configurations. To ensure fairness, reproducibility and controller comparison, three prompts of increasing specificity were designed and applied consistently across all 80 configurations. These prompts were referenced as Broad, Mid-Level and Specific. Each prompt was fixed word-for-word and reused in every test case, eliminating prompt variability as a source of bias. Outputs were recorded manually to allow for human verification of correctness, with the evaluation focusing exclusively on the accuracy of misconfiguration detection.

### *3.4.1 Broad Assessment*

The first test case adopted a broad, open-ended approach designed to establish a baseline of performance. The prompt framed the model as a network security analyst reviewing a router configuration and asked it to determine whether the configuration was secure. It was instructed to flag any security issues, misconfigurations, Mistype errors, or best practice violations without reference to any standards. If no issues were present, the model was required to state explicitly that the configuration was secure.

> You are a network security analyst reviewing a router configuration for deployment in a mid-sized corporate network. Task: Is this configuration secure? Please identify any security issues, misconfigurations, typos, or best practice violations. If the configuration is secure, state: ”Yes, this configuration appears secure.” Otherwise, explain the issues briefly.

***Figure 3.10:*** *Broad Assessment Prompt*

The Broad prompt was deliberately constructed in a short and simple form. This minimised the risk of misinterpretation and avoided providing additional cues that could bias the model. By keeping the instruction minimal, the test measured the extent to which GPT could rely on its generalised knowledge of networking practices without external benchmarks.

### *3.4.2 Mid-Level-Assessment*

The second test case introduced a stronger compliance focus by referencing the CIS Cisco IOS 15 Benchmarks directly in the prompt. However, no benchmark excerpts were provided. Instead, the model was required to rely on its internalised knowledge of CIS standards from training data. This case tested whether GPT could recall and apply compliance driven reasoning when instructed, without being given the official benchmark text.

> You are a network security analyst reviewing a router configuration for deployment in a mid-sized corporate network. Task: According to CIS Benchmarks is this configuration secure? Please identify any security issues, misconfigurations, typos, or best practice violations. If the configuration is secure, state: ”Yes, this configuration appears secure.” Otherwise, explain the issues briefly.

***Figure 3.11:*** *Mid-Level Assessment Prompt*

The Mid prompt was constructed by taking the Broad prompt and explicitly adding a reference to the CIS Benchmarks. This small modification was intended to activate benchmark-oriented reasoning within the model, encouraging it to produce more structured and compliance focused evaluations.

### *3.4.3 Specific, CIS-Guided Assessment*

The third test case was the most constrained and prescriptive. In this case, the model was instructed to review only a single protocol (AAA, EIGRP, OSPF or RIP) within the configuration and to assess compliance solely against the corresponding section of the CIS Benchmarks. Unlike the Mid-level tests, full-protocol specific excerpts of the official CIS IOS 15 Benchmarks were attached to the prompt as external files.

> You are a network security analyst reviewing ONLY an OSPF configuration for deployment in a mid-sized corporate network. Assess whether this configuration complies with the attached CIS Benchmarks for OSPF security. Identify any security issues, misconfigurations, typos, or violations of CIS best practices. If the configuration meets the attached CIS requirements, state: 'Yes, this OSPF configuration appears secure and CIS-compliant.' Otherwise, briefly list the issues and how they deviate from CIS guidelines.

***Figure 3.12:*** *Specific Assessment Prompt*

The Specific prompt was carefully constructed to limit the model's scope to a single protocol and to force reliance on authoritative rules. By explicitly providing the benchmark text, the prompt tested whether GPT could correctly interpret and apply prescriptive compliance requirements, simulating the conditions of a professional security audit. In contrast to the Mid and Broad tests, the Specific prompt improved the structure and consistency of responses by providing CIS Benchmark excerpts. This allowed the model to anchor its reasoning on authoritative standards rather than relying solely on internal training knowledge, reducing variability and limiting hallucinated or omitted findings.

Together, these three test cases formed a progression of guidance, the Broad case tested open ended reasoning without standards, the Mid case tested whether referencing CIS Benchmarks improved performance through internalised knowledge, and the Specific case evaluated the model's ability to apply explicit compliance rules. This layered design provided a comprehensive evaluation of LLM performance across general reasoning, implicit knowledge and explicit compliance checking.

### *3.4.4 Standardisation and Reproducibility*

To maintain consistency, all three prompts were applied word-for-word across every configuration without modification. This prevented prompt variation from influencing results and ensured t hat differences could be attributed to the level of specificity. Early testing confirmed that running the same prompt multiple times produced similar outputs, so each configuration was evaluated once per prompt.

All outputs were recorded manually rather than through automated pipelines. This approach enabled direct verification of correctness and clear categorisation of detections as correct, partial or incorrect. Manual inspection also provided additional context, such as observing when the models compared their findings to CIS standards or misclassified valid commands, offering insights that a purely automated accuracy score would not capture.

## 3.5 Evaluation Procedure

The evaluation procedure defined how the dataset, prompts and models were combined to assess the ability of GPT-4o to detect misconfigurations and Mistype errors. This section outlines the execution of test cases, the process of recording results, the criteria for judging correctness and the measures taken to ensure reproducibility.

### *3.5.1 Execution of Test Cases*

Each of the 80 configurations in the dataset were tested once using the three different prompt types, all testing and was performed using OpenAI's GPT-4o model.

Prompts were always presented before the configuration, ensuring that the model understood its assigned role before processing the technical content. For Broad and Mid cases, configurations were attaches as .txt files, while in the Specific case they were copy-pasted directly into the prompt alongside the CIS excerpts. This distinction was necessary because GPT occasionally misinterpreted the volume of information when multiple files were attached. Each run was performed in a temporary chat session that was reset after every configuration, ensuring that no conversational memory was carried over between tests.

## 3.6 Analysis Methods

The purpose of the analysis was to interpret the outputs generated by GPT-4o in a systematic way that linked back to the research questions. Since the evaluation procedure produced a large number of raw responses, it was necessary to apply structured methods to measure detection accuracy and compare performance across prompts and protocols. The analysis combined quantitative accuracy scores with comparative visualisations to highlight patterns in the model's behaviour.

### *3.6.1 Quantitative Analysis*

The primary metric applied was the Perfect Predictions (PP), which classifies an LLM-generated security assessment as correct only if it matched the expected output. The PP Score was calculated by dividing the number of errors correctly identified by the number of errors deliberately injected into each configuration [12].

$$PP = \frac{\text{Number of samples with a perfect match}}{\text{Total number of samples}}$$

This method was chosen as it allowed the analysis to account for both complete detections and partial detections (for example identifying one error out of three). By capturing partial accuracy, PP Score provided a more representative measure than a simple binary correct/incorrect classification. PP Scores were calculated at three levels:

a) Total dataset accuracy, expressed separately for misconfiguration and Mistype errors. Misconfigurations were benchmark-driven violations, while Mistype errors represented general robustness checks.

b) Per-protocol accuracy, covering AAA, EIGRP, OSPF and RIP. This enabled the analysis to highlight whether some protocols presented greater challenges for the models.

c) Per-prompt accuracy, with separate scores for Broad, Mid and Specific prompts, showing how levels of prompt specificity influenced detection rates.

### 3.6.2 Comparative Analysis

The second stage of analysis focused on comparing the accuracy of the three test cases. Broad, Mid and Specific prompts were treated as independent test cases, each evaluated against the same 80 configurations. Comparisons were therefore made between prompt types, rather than between runs of the same prompt. This design choice ensured that differences in results could be directly attributed to the level of guidance given to the model.

To present these comparisons clearly, results were visualised using bar charts showing protocol-level PP Scores for each prompt type, alongside aggregated totals for misconfigurations and Mistype errors.

# Chapter 4: Analysis of Results and Discussion

## 4.1 Introduction of the Analysis and Discussion

This research hypothesised that large language models (LLMs) could be used to detect security misconfigurations in Cisco IOS configurations when assessed against the CIS Benchmarks. By employing a structured evaluation framework built on controlled datasets and carefully engineered prompts, the study aimed to measure both the accuracy and reliability of GPT-4o in identifying benchmark-aligned errors as well as common configuration mistakes. The primary objective was to determine whether prompt engineering could be used to guide the models toward more consistent and accurate compliance assessments. As part of this investigation, model behaviour was also examined under different temperature settings, Increasing the temperature often led to hallucinations, where the model fabricated information, added commands that were never specified or ignored key instructions altogether, Conversely, lowering the temperature made the model overly rigid, producing outputs that strictly followed the instructions but frequently inserted "UNKNOWN" placeholders where variation or contextual adaptation was required. These observations highlighted that some degree of randomness is necessary for balanced performance, as overly deterministic outputs limited completeness, while excessive randomness compromised reliability. Additionally, the research aimed to examine how varying levels of prompt specificity (Broad, Mid and Specific) shaped the model's performance, and to compare the outcomes across multiple protocol domains. This study also aimed to capture qualitative patterns in the model's responses, such as recurring misclassifications or hallucinations, to better understand their interpretative strengths and limitations. In the following sections, the analysis will present the results of these evaluations, compare model performance across prompt types and protocols, and critically discuss the implications of these findings for the integration of LLMs into professional network auditing and compliance workflows.

## 4.2 Presentation of Findings

### *4.2.1 Test Case 1: Broad Prompt*

The Broad prompt was designed as the baseline test, requiring the model to review configurations without explicit reference to the CIS Benchmarks. Its purpose was to measure GPT's ability to apply general networking knowledge when identifying misconfigurations and Mistype errors. The results indicated limited effectiveness in detecting benchmark aligned issues, with stronger but still inconsistent performance in identifying simple typographical errors.

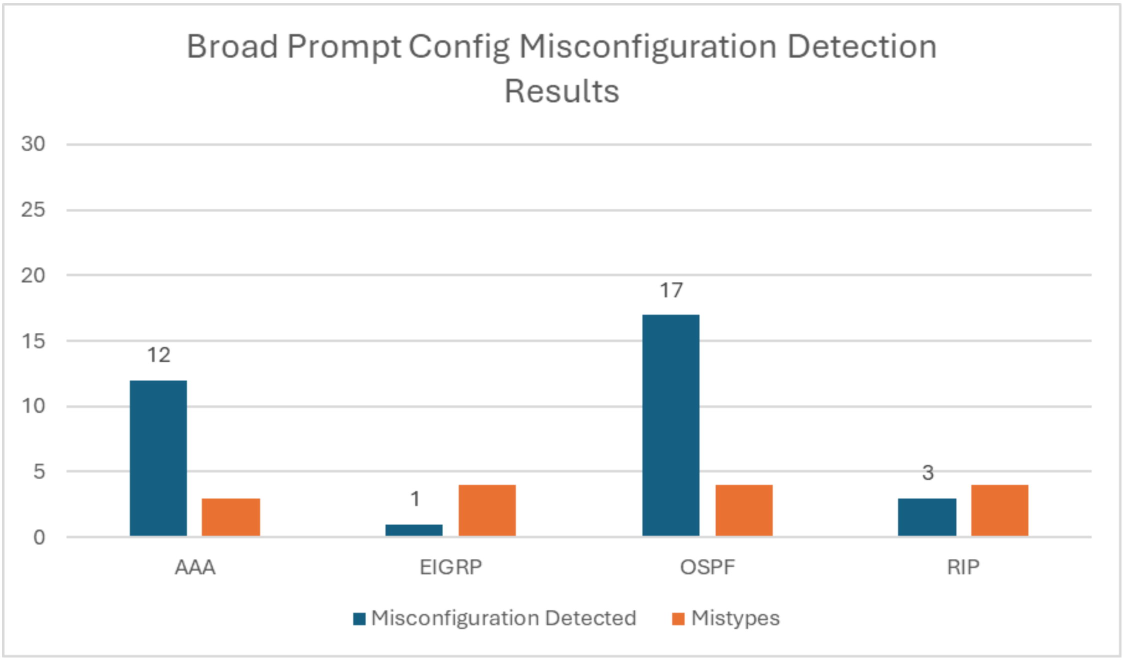


***Figure 4.1:*** *Broad Prompt Misconfiguration Detection Results*

| AAA | | | |
|---|---|---|---|
| **Number of Errors Present** | **1 Error Detected** | **2 Errors Detected** | **3 Errors Detected** |
| 1 | 5/5 | – | – |
| 2 | 1/5 | 1/5 | – |
| 3 | 4/5 | 0/5 | 0/5 |
| | | | **PP Score: 40%** |
| **EIGRP** | | | |
| 1 | 0/5 | | |
| 2 | 1/5 | 0/5 | |
| 3 | 0/5 | 0/5 | 0/5 |
| | | | **PP Score: 3%** |
| **OSPF** | | | |
| 1 | 1/5 | | |
| 2 | 2/5 | 3/5 | |
| 3 | 1/5 | 2/5 | 1/5 |
| | | | **PP Score: 56%** |
| **RIP** | | | |
| 1 | 0/5 | | |
| 2 | 2/5 | 0/5 | |
| 3 | 1/5 | 0/5 | 0/5 |
| | | | **PP Score: 10%** |
| | | | **Total PP Score: 28%** |

***Table 4.1:*** *Broad Prompt Individual Results*

| Protocol | Mistypes Not Detected | Mistypes Detected | PP Score |
|---|---|---|---|
| **AAA** | **2/5** | **3/5** | **60%** |
| **EIGRP** | **1/5** | **4/5** | **80%** |
| **OSPF** | **1/5** | **4/5** | **80%** |
| **RIP** | **1/5** | **4/5** | **80%** |
| | | | **Total PP Score: 75%** |

***Table 4.2:*** *Broad Prompt Mistype Detection Results*

Chart 4.1 and Tables 1 summarises the results across all four protocols. Out of 120 benchmark-related errors, only 33 were detected, resulting in a total PP Score of 28%. Accuracy varied significantly by protocol. OSPF achieved the highest PP Score of 56%, followed by AAA at 40%, while RIP and especially EIGRP performed poorly, with scores of 10% and 3% respectively. By contrast, Mistype errors were detected resulting in a total PP Score of 75%. Accuracy varied between protocol with EIGRP, OSPF and RIP all having PP Scores of 80% while AAA scored a PP Score of 60%. This indicated that GPT was more reliable at identifying basic syntactic mistakes than at enforcing benchmark compliance.

| **AAA Protocol Errors** | |
|---|---|
| **Errors** | **Errors Detected** |
| Service password-encryption | 3/3 |
| Enable secret | 1/1 |
| Aaa accounting system | 3/5 |
| Aaa authorization exec | 2/3 |
| Aaa authorization reverse-access | 0/4 |
| Aaa accounting commands 15 | 0/1 |
| Aaa new-model | 0/2 |
| Aaa authorization config-commands | 1/2 |
| Aaa authorization network | 0/2 |
| Line vty | 2/2 |
| **EIGRP Protocol Errors** | |
| **Errors** | **Errors Detected** |
| Router eigrp | 0/2 |
| Key-chain | 0/4 |
| Key | 0/3 |
| Key-string | 0/1 |
| Ip authentication mode eigrp md5 | 0/4 |
| Ip authentication key-chain eigrp | 1/9 |
| Passive-interface | 0/7 |
| **OSPF Protocol Errors** | |
| **Errors** | **Errors Detected** |
| Router ospf | 1/2 |
| Area authentication message-digest | 5/9 |
| Ip ospf message-digest-key | 10/19 |
| **RIP Protocol Errors** | |
| **Errors** | **Errors Detected** |
| Router rip | 0/4 |
| Version 2 | 0/2 |
| Ip rip authentication mode md5 | 0/4 |
| Ip rip authentication key-chain | 0/4 |
| Key-chain | 1/3 |
| Network | 0/4 |
| Key-string | 1/1 |
| Passive-interface | 0/2 |
| Redistribute | 0/4 |
| Maximum-paths | 0/3 |
| Offset-list | 0/1 |

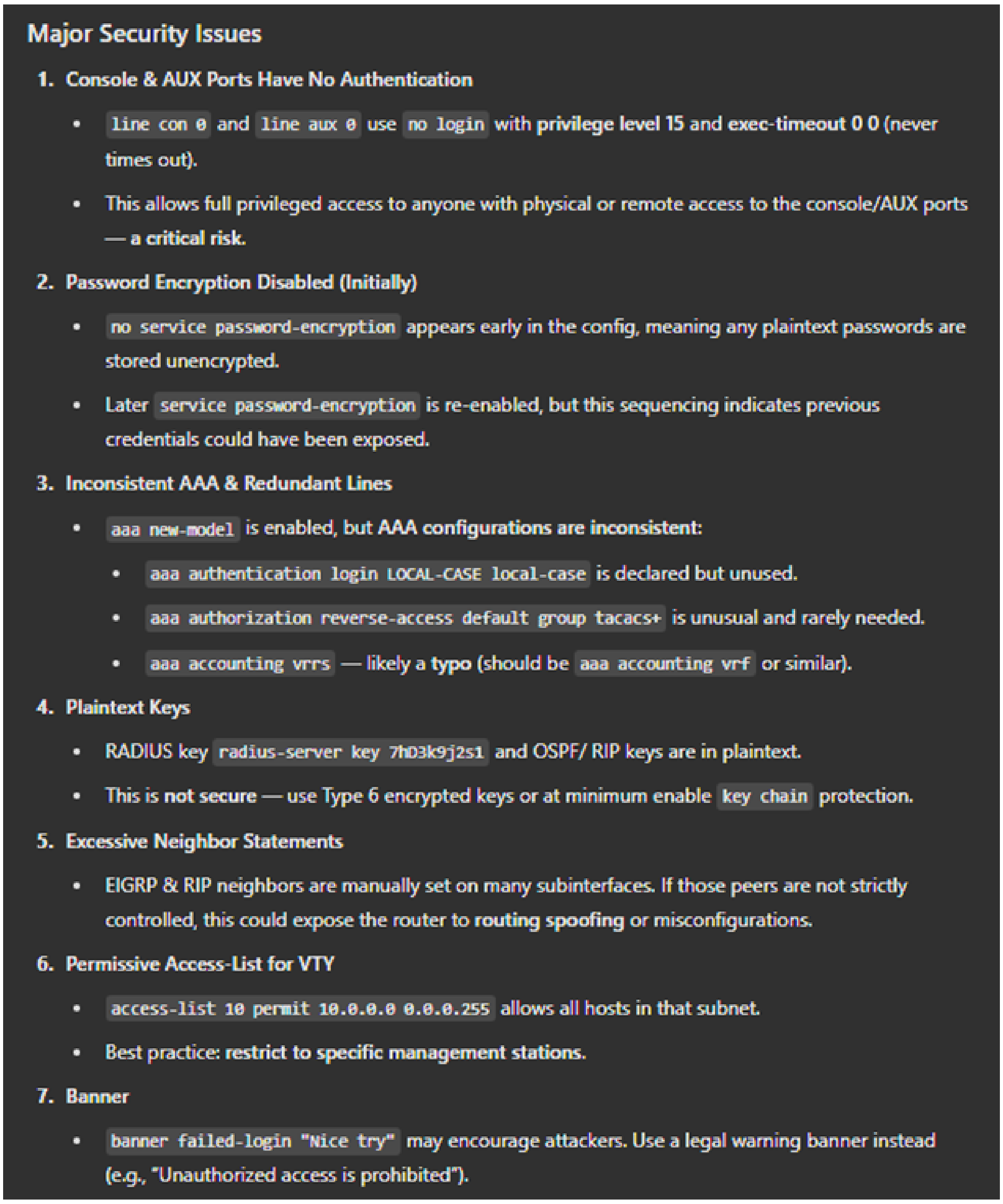
***Table 4.3:*** *Broad Error Detections*

To better understand recurring strengths and weaknesses, the configurations were further analysed to identify specific requirements that were either consistently detected or frequently overlooked. Table 3 shows that certain fundamental commands, “service password-encryption” in AAA was always detected with a score of 3/3 detections and “ip ospf message-digest-key md5” in OSPF with a score of 10/19 detections, were reliably flagged when missing, suggesting that the model could recognise high-level authentication mechanisms and simple per-interface security features. EIGRP and RIP had constantly not detected misconfigurations in these protocols for this test. Table 3 also highlights critical requirements that the model struggled with, “aaa authorization reverse-access” in AAA is one of them with a score of 0/4 detections and “area authentication message-digest” in OSPF with a score of 5/9 detections. These are more structural, policy-driven requirements, and the model often failed to detect their absence despite their central importance to CIS compliance. EIGRP and RIP had too low of detection rates to produce meaningful patterns.

Another type of analysis that was done considered the references that GPT cited when justifying its outputs. To explore this, three configurations from each protocol were sampled, and all references mentioned in the model’s reasoning were documents. As shown in Table 4, GPT frequently cited Cisco documentation, CIS Benchmarks and NIST guidelines, occasionally drawing on references such as NSA, RFC standards, DISA STIGs or real-world application examples. While the diversity of references suggests exposure to authoritative material, their use was inconsistent and not always aligned with the errors under review. Cisco, CIS and NIST were by far the most commonly used, indicating that GPT anchored its reasoning on broadly recognised security sources, but without the precision required for consistent compliance auditing.

| Reference Source | Count |
|---|---|
| **Cisco References** | **12/12** |
| **CIS References** | **11/12** |
| **NIST References** | **10/12** |
| **Real-world applications Referenced** | **5/12** |
| **NSA References** | **2/12** |
| **RFC References** | **2/12** |
| **DISA STIG References** | **1/12** |

***Table 4.4:*** *Most common references used by Broad Prompt*



## Major Security Issues

1. **Console & AUX Ports Have No Authentication**
   - `line con 0` and `line aux 0` use `no login` with **privilege level 15** and **exec-timeout 0 0** (never times out).
   - This allows full privileged access to anyone with physical or remote access to the console/AUX ports — **a critical risk.**
2. **Password Encryption Disabled (Initially)**
   - `no service password-encryption` appears early in the config, meaning any plaintext passwords are stored unencrypted.
   - Later `service password-encryption` is re-enabled, but this sequencing indicates previous credentials could have been exposed.
3. **Inconsistent AAA & Redundant Lines**
   - `aaa new-model` is enabled, but **AAA configurations are inconsistent:**
     - `aaa authentication login LOCAL-CASE local-case` is declared but unused.
     - `aaa authorization reverse-access default group tacacs+` is unusual and rarely needed.
     - `aaa accounting vrrs` — likely a **typo** (should be `aaa accounting vrf` or similar).
4. **Plaintext Keys**
   - RADIUS key `radius-server key 7hD3k9j2s1` and OSPF/ RIP keys are in plaintext.
   - This is **not secure** — use Type 6 encrypted keys or at minimum enable `key chain` protection.
5. **Excessive Neighbor Statements**
   - EIGRP & RIP neighbors are manually set on many subinterfaces. If those peers are not strictly controlled, this could expose the router to **routing spoofing** or misconfigurations.
6. **Permissive Access-List for VTY**
   - `access-list 10 permit 10.0.0.0 0.0.0.255` allows all hosts in that subnet.
   - Best practice: **restrict to specific management stations.**
7. **Banner**
   - `banner failed-login "Nice try"` may encourage attackers. Use a legal warning banner instead (e.g., "Unauthorized access is prohibited").

***Figure 4.2:*** *Broad Prompt Output Example*

In terms of qualitative patterns, the model frequently produced short and list-like responses as shown in Figure 1 rather than detailed checklists. While it often flagged certain recurring issues, it consistently failed to detect critical requirements in protocols as shown in Table 7. Overall, the Broad prompt demonstrated the weakness of unguided prompting. While capable of identifying obvious typos and occasional misconfigurations, its detection of benchmark-aligned errors was inconsistent and unreliable across protocols. These findings establish a baseline for comparison with the Mid and Specific prompts, where addition guidance was introduced.

## 4.3 Test Case 2: Mid Prompt

The Mid prompt introduced explicit reference to the CIS Benchmarks but did not provide excerpts of the standards themselves. Instead, the model was instructed to review the configurations “according to CIS Benchmarks”, requiring them to rely on any embedded knowledge of the framework gained during training. This test case therefore measured whether mentioning the CIS Benchmarks within the prompt alone improved detection accuracy compared to a purely unguided assessment.

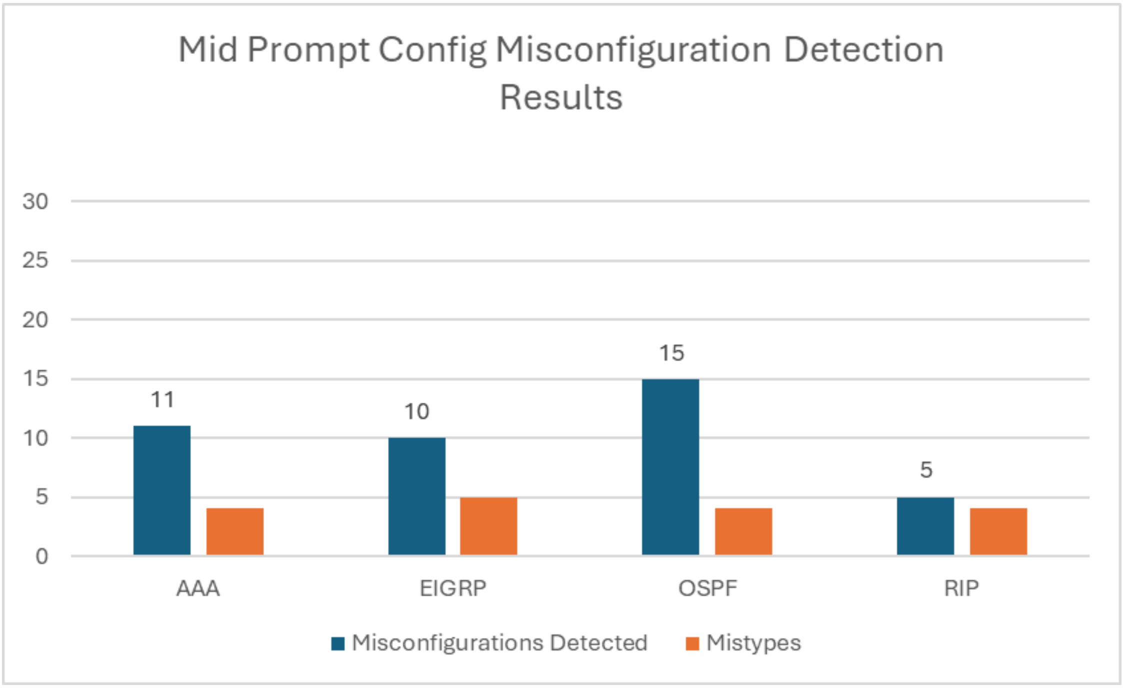


***Figure 4.3:*** *Mid Prompt Misconfiguration Detection Results*

| **AAA** | | | |
|---|---|---|---|
| **Number of Errors Present** | **1 Error Detected** | **2 Errors Detected** | **3 Errors Detected** |
| 1 | 4/5 | – | – |
| 2 | 3/5 | 1/5 | – |
| 3 | 2/5 | 1/5 | 0/5 |
| | | | **PP Score: 37%** |
| **EIGRP** | | | |
| 1 | 1/5 | | |
| 2 | 3/5 | 0/5 | |
| 3 | 1/5 | 2/5 | 0/5 |
| | | | **PP Score: 30%** |
| **OSPF** | | | |
| 1 | 1/5 | | |
| 2 | 2/5 | 2/5 | |
| 3 | 1/5 | 1/5 | 2/5 |
| | | | **PP Score: 53%** |
| **RIP** | | | |
| 1 | 1/5 | | |
| 2 | 2/5 | 1/5 | |
| 3 | 0/5 | 0/5 | 0/5 |
| | | | **PP Score: 17%** |
| | | | **Total PP Score: 33%** |

| Protocol | Mistypes Not Detected | Mistypes Detected | PP Score |
|---|---|---|---|
| **AAA** | **1/5** | **4/5** | **60%** |
| **EIGRP** | **0/5** | **5/5** | **100%** |
| **OSPF** | **1/5** | **4/5** | **80%** |
| **RIP** | **1/5** | **4/5** | **80%** |
| | | | **Total PP Score: 80%** |

***Table 4.6:*** *Mid Prompt Mistype Detection Results*

The results showed moderate gains in benchmark-related error detection as shown in Table 5. Out of 120 misconfigurations implemented, 39 were identified, resulting in a total PP Score of 33%. Accuracy differed by protocol, with OSPF achieving the highest PP Score of 53% followed by AAA at 37% and EIGRP at 30%. RIP remained the weakest domain with only 17% of errors detected. Mistype errors were handled more effectively than in the Broad test, with a PP Score of 85% across all protocols. Protocol-specific mistype scores ranged from 80% in AAA, OSPF and RIP to 100% in EIGRP, demonstrating strong consistency in identifying basic syntactic issues. A per-protocol breakdown of misconfigurations detected is presented in Chart 4.2. As with the Broad case, further analysis was carried out to determine recurring strengths and weaknesses. Table 14 highlights commands that the Mid prompt reliably detected. These included "service password-encryption" in AAA with 3/3 detections, "key chain" in EIGRP with 3/4 detections, "ip ospf message-difest-key md5" in OSPF with 13/18 detections and "ip rip authentication key-chain" in RIP with 2/2 detections. These results suggest that the introduction of benchmark framing improved recognition of certain authentication and encryption requirements, especially within

OSPF.

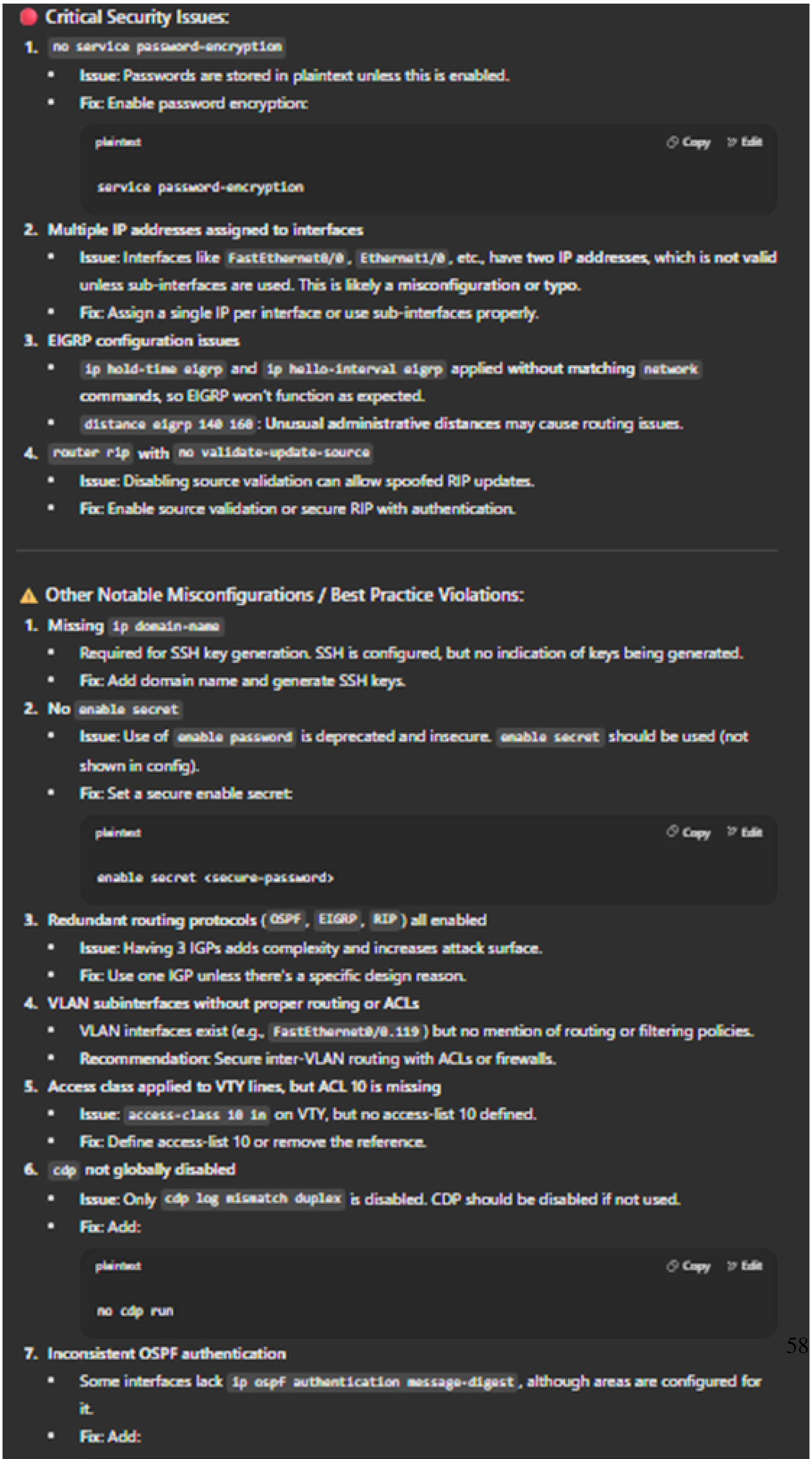
| **AAA Protocol Errors** | |
|---|---|
| **Errors** | **Errors Detected** |
| Service password-encryption | 3/3 |
| Enable secret | 1/1 |
| Aaa accounting system | 0/5 |
| Aaa authorization exec | 3/3 |
| Aaa authorization reverse-access | 3/4 |
| Aaa accounting commands 15 | 0/1 |
| Aaa new-model | 0/2 |
| Aaa authorization config-commands | 1/2 |
| Aaa authorization network | 0/2 |
| Line vty | 2/2 |
| **EIGRP Protocol Errors** | |
| **Errors** | **Errors Detected** |
| Router eigrp | 0/2 |
| Key-chain | 1/4 |
| Key | 2/3 |
| Key-string | 0/1 |
| Ip authentication mode eigrp md5 | 2/4 |
| Ip authentication key-chain eigrp | 3/9 |
| Passive-interface | 1/7 |
| **OSPF Protocol Errors** | |
| **Errors** | **Errors Detected** |
| Router ospf | 0/2 |
| Area authentication message-digest | 5/9 |
| Ip ospf message-digest-key | 11/19 |
| **RIP Protocol Errors** | |
| **Errors** | **Errors Detected** |
| Router rip | 0/4 |
| Version 2 | 0/2 |
| Ip rip authentication mode md5 | 1/4 |
| Ip rip authentication key-chain | 2/4 |
| Key-chain | 1/3 |
| Network | 0/4 |
| Key-string | 1/1 |
| Passive-interface | 0/2 |
| Redistribute | 0/4 |
| Maximum-paths | 0/3 |
| Offset-list | 0/1 |

***Table 4.7:*** *Mid Error Detections*

However, as shown in Table 7, the model continued to overlook several critical requirements. For AAA, "aaa accounting system" was inconsistently flagged with 0/5 detections, EIGRP exhibited weak detection of "passive interface", with a score of 1/7 detections. In OSPF, the omission of "router ospf" was overlooked more often than not with a score of 0/2 and in RIP, missing "network" statements were consistently undetected with a score of 0/4. These gaps demonstrate that while Mid prompting increased structure and alignment with compliance concepts, reliability in detecting mandatory benchmark rules remained inconsistent Finally, an examination of the references cited by GPT in its explanations in Table 8 provided further insight into how it justified its outputs. Three configurations per protocol were sampled, and citations were documented. Cisco and CIS documentation were the most frequently invoked with them being used 12 times, following by NIST guidelines with 8 times, real-world application examples with 6 and NSA references with 4. The consistent appearance of Cisco and CIS sources suggest that the Mid prompt successfully anchored GPT's reasoning to widely recognised compliance frameworks. However, the uneven use of supporting sources illustrates why detection accuracy remained inconsistent.

| Reference Source | Count |
|---|---|
| **Cisco References** | **12/12** |
| **CIS References** | **12/12** |
| **NIST References** | **8/12** |
| **Real-world applications Referenced** | **6/12** |
| **NSA References** | **4/12** |

***Table 4.8:*** *Most common references used by Mid Prompt*



🔴 **Critical Security Issues:**

1. **`no service password-encryption`**
   - Issue: Passwords are stored in plaintext unless this is enabled.
   - Fix: Enable password encryption:

     ```plaintext
     service password-encryption
     ```

2. **Multiple IP addresses assigned to interfaces**
   - Issue: Interfaces like `FastEthernet0/0`, `Ethernet1/0`, etc., have two IP addresses, which is not valid unless sub-interfaces are used. This is likely a misconfiguration or typo.
   - Fix: Assign a single IP per interface or use sub-interfaces properly.
3. **EIGRP configuration issues**
   - `ip hold-time eigrp` and `ip hello-interval eigrp` applied without matching `network` commands, so EIGRP won't function as expected.
   - `distance eigrp 140 160`: Unusual administrative distances may cause routing issues.
4. **`router rip` with `no validate-update-source`**
   - Issue: Disabling source validation can allow spoofed RIP updates.
   - Fix: Enable source validation or secure RIP with authentication.

---

⚠️ **Other Notable Misconfigurations / Best Practice Violations:**

1. **Missing `ip domain-name`**
   - Required for SSH key generation. SSH is configured, but no indication of keys being generated.
   - Fix: Add domain name and generate SSH keys.
2. **No `enable secret`**
   - Issue: Use of `enable password` is deprecated and insecure. `enable secret` should be used (not shown in config).
   - Fix: Set a secure enable secret:

     ```plaintext
     enable secret <secure-password>
     ```

3. **Redundant routing protocols (`OSPF`, `EIGRP`, `RIP`) all enabled**
   - Issue: Having 3 IGPs adds complexity and increases attack surface.
   - Fix: Use one IGP unless there's a specific design reason.
4. **VLAN subinterfaces without proper routing or ACLs**
   - VLAN interfaces exist (e.g., `FastEthernet0/0.119`) but no mention of routing or filtering policies.
   - Recommendation: Secure inter-VLAN routing with ACLs or firewalls.
5. **Access class applied to VTY lines, but ACL 10 is missing**
   - Issue: `access-class 10 in` on VTY, but no access-list 10 defined.
   - Fix: Define access-list 10 or remove the reference.
6. **`cdp` not globally disabled**
   - Issue: Only `cdp log mismatch duplex` is disabled. CDP should be disabled if not used.
   - Fix: Add:

     ```plaintext
     no cdp run
     ```

7. **Inconsistent OSPF authentication**
   - Some interfaces lack `ip ospf authentication message-digest`, although areas are configured for it.
   - Fix: Add:

Overall, the Mid Prompt demonstrated that referencing benchmarks within the instructions improves alignment with compliance-oriented reasoning and enhances detection of certain classes of misconfigurations. Nonetheless, accuracy gains were limited and several benchmark-critical omissions continued to be missed across all protocols.

### *4.3.1 Test Case 3: Specific Prompt*

The Specific prompt represented the most constrained test case, limiting the model's evaluation to a single protocol and attaching the corresponding section of the CIS Benchmark for reference. By providing explicit standards alongside the configuration, this test case measured the model's ability to apply authoritative compliance rules within a defined scope.

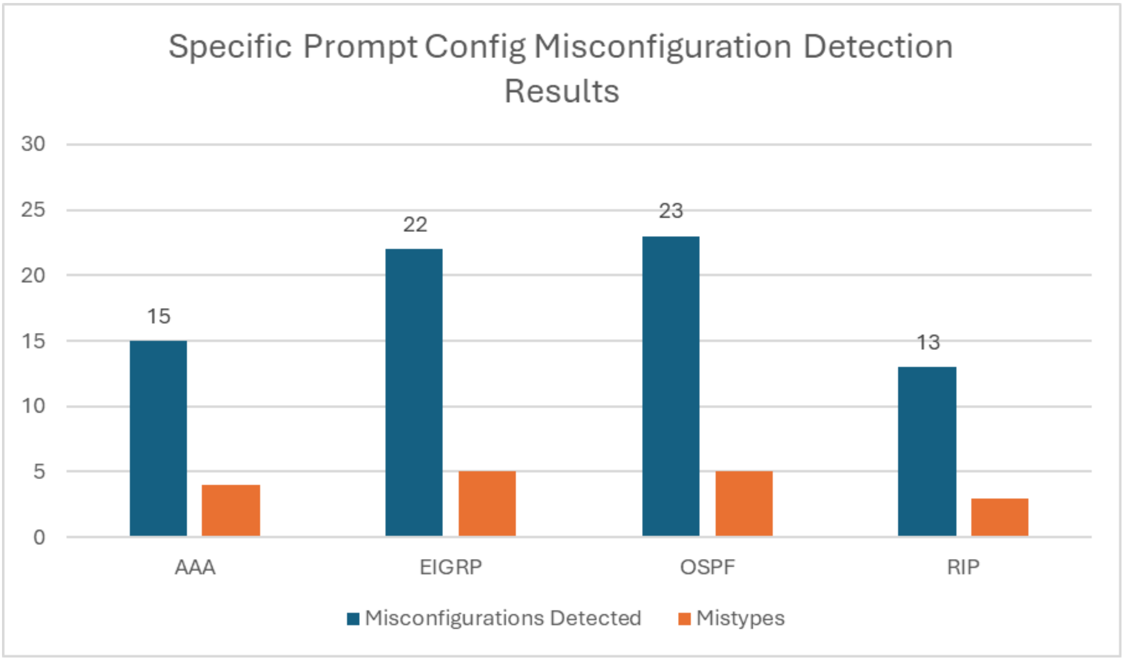


***Figure 4.5:*** *Specific Prompt Misconfiguration Detection Results*

| AAA | | | |
|---|---|---|---|
| **Number of Errors Present** | **1 Error Detected** | **2 Errors Detected** | **3 Errors Detected** |
| 1 | 5/5 | – | – |
| 2 | 1/5 | 2/5 | – |
| 3 | 3/5 | 1/5 | 0/5 |
| | | | **PP Score: 50%** |
| **EIGRP** | | | |
| 1 | 2/5 | | |
| 2 | 1/5 | 4/5 | |
| 3 | 1/5 | 2/5 | 2/5 |
| | | | **PP Score: 73%** |
| **OSPF** | | | |
| 1 | 3/5 | | |
| 2 | 3/5 | 2/5 | |
| 3 | 0/5 | 2/5 | 2/5 |
| | | | **PP Score: 77%** |
| **RIP** | | | |
| 1 | 2/5 | | |
| 2 | 4/5 | 1/5 | |
| 3 | 3/5 | 1/5 | 0/5 |
| | | | **PP Score: 43%** |
| | | | **Total PP Score: 61%** |

***Table 4.9:*** *Specific Prompt Individual Results*

| Protocol | Mistypes Not Detected | Mistypes Detected | PP Score |
|---|---|---|---|
| **AAA** | **1/5** | **4/5** | **80%** |
| **EIGRP** | **0/5** | **5/5** | **100%** |
| **OSPF** | **0/5** | **5/5** | **100%** |
| **RIP** | **2/5** | **3/5** | **60%** |
| | | | **Total PP Score: 85%** |

***Table 4.10:*** *Specific Prompt Mistype Detection Results*

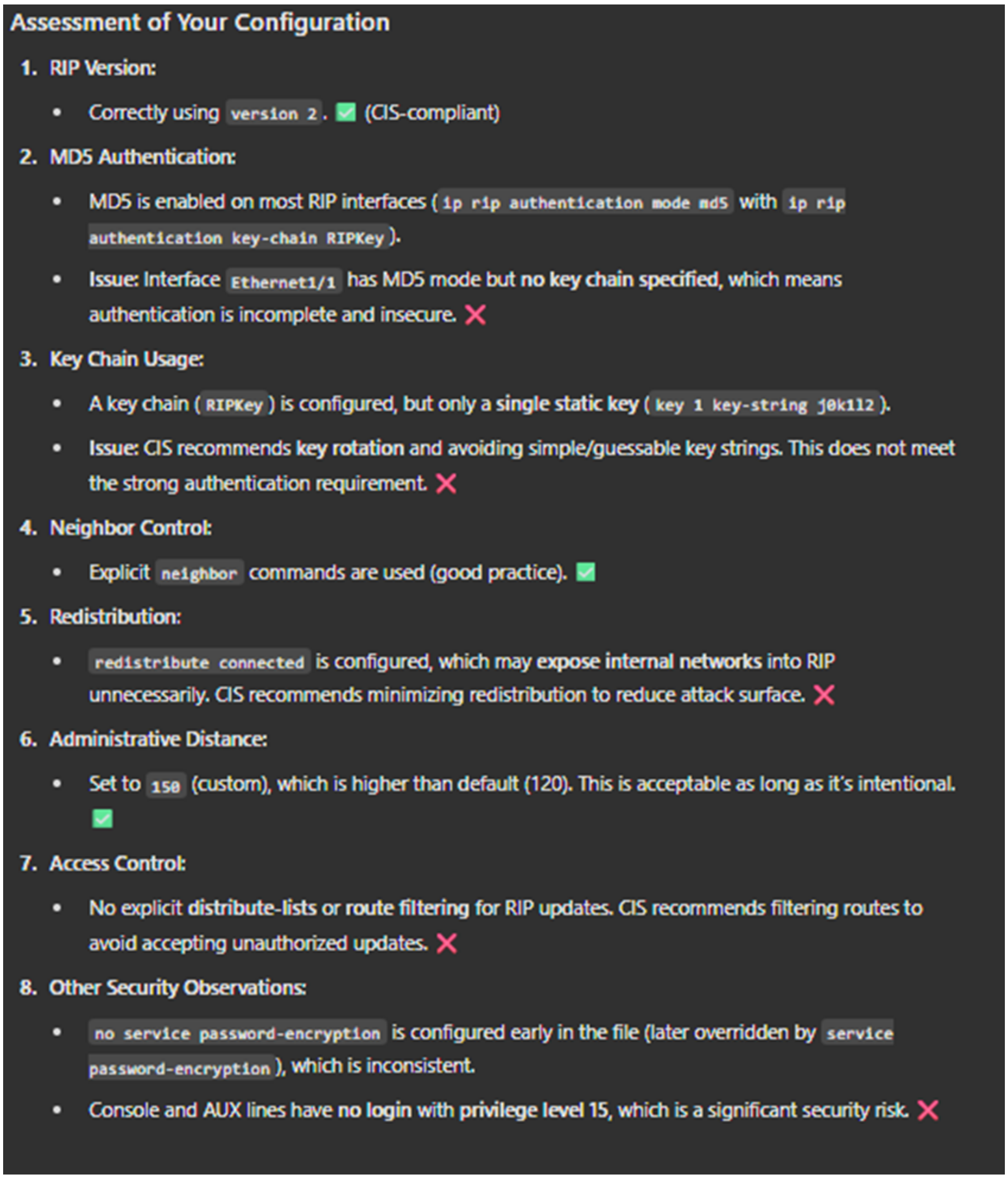
The results shown in Tables 9 demonstrated considerably higher accuracy than the previous test cases. Out of 120 benchmark-related errors, 73 were detected, giving a total PP Score of 61%. Protocol-level performance varied, with OSPF achieving the highest accuracy of 77%, followed by EIGRP at 73%, AAA at 50% and RIP at 43%. Mistype detection was strong overall as shown in Table 10, with a combined PP Score of 85%, though individual protocol scores ranged from 60% in RIP to 100% in both EIGRP and OSPF. A per-protocol breakdown of misconfigurations detected is illustrated in Chart 4.3.

| **AAA Protocol Errors** | |
|---|---|
| **Errors** | **Errors Detected** |
| Service password-encryption | 3/3 |
| Enable secret | 1/1 |
| Aaa accounting system | 4/5 |
| Aaa authorization exec | 2/3 |
| Aaa authorization reverse-access | 0/4 |
| Aaa accounting commands 15 | 1/1 |
| Aaa new-model | 1/2 |
| Aaa authorization config-commands | 1/2 |
| Aaa authorization network | 1/2 |
| Line vty | 1/2 |
| **EIGRP Protocol Errors** | |
| **Errors** | **Errors Detected** |
| Router eigrp | 0/2 |
| Key-chain | 4/4 |
| Key | 2/3 |
| Key-string | 0/1 |
| Ip authentication mode eigrp md5 | 3/4 |
| Ip authentication key-chain eigrp | 9/9 |
| Passive-interface | 4/7 |
| **OSPF Protocol Errors** | |
| **Errors** | **Errors Detected** |
| Router ospf | 1/2 |
| Area authentication message-digest | 7/9 |
| Ip ospf message-digest-key | 15/19 |
| **RIP Protocol Errors** | |
| **Errors** | **Errors Detected** |
| Router rip | 2/4 |
| Version 2 | 0/2 |
| Ip rip authentication mode md5 | 4/4 |
| Ip rip authentication key-chain | 2/4 |
| Key-chain | 2/3 |
| Network | 0/4 |
| Key-string | 1/1 |
| Passive-interface | 2/2 |
| Redistribute | 0/4 |
| Maximum-paths | 0/3 |
| Offset-list | 0/1 |

***Table 4.11:*** *Specific Error Detections*

Further inspection of detection patterns is summarised in Table 11, which highlights the key requirements that were consistently identified. For AAA “service password-encryption” in AAA was detected in all 3/3 cases, “ip authentication key-chain eigrp” was identified in 9/9 runs, “ip ospf message-digest key” in OSPF was flagged in 15/19 cases and “ip rip authentication mode md5” in RIP was always detected with a score of 4/4. These results confirm that providing benchmark excerpts helped the model anchor to certain critical CIS requirements more reliably. Nevertheless, certain benchmark-mandated rules were still missed, as shown in Table 11. In AAA, “aaa authorization reverse-access” was overlooked in all 0/4 cases. RIP was particularly weak, with “network” consistently being missed in all 4 cases. No consistently missed requirements were observed in EIGRP and OSPF. These results highlight that even under conditions when authoritative CIS excerpts were supplied, GPT was not uniformly consistent in applying all rules.

Qualitative analysis of the outputs revealed significant improvements compared to the Broad and Mid prompts. Responses where checklist-like, more structured and explicitly referenced CIS Benchmarks, as illustrated in Figure 3. Explanations were generally more detailed, precise and audit-oriented, resembling the style of professional compliance reviews. This indicated that grounding GPTs reasoning in benchmark excerpts reduced ambiguity and encouraged outputs that aligned more closely with compliance auditing practices. However, analysis of GPT’s cited references shows that it did not exclusively rely on the CIS Benchmarks even when they were explicitly attached. As shown in Table 24, three configurations were



Assessment of Your Configuration

1. **RIP Version:**
   - Correctly using `version 2`. ✅ (CIS-compliant)
2. **MD5 Authentication:**
   - MD5 is enabled on most RIP interfaces (`ip rip authentication mode md5` with `ip rip authentication key-chain RIPKey`).
   - **Issue:** Interface `Ethernet1/1` has MD5 mode but **no key chain specified**, which means authentication is incomplete and insecure. ❌
3. **Key Chain Usage:**
   - A key chain (`RIPKey`) is configured, but only a **single static key** (`key 1 key-string j0k112`).
   - **Issue:** CIS recommends **key rotation** and avoiding simple/guessable key strings. This does not meet the strong authentication requirement. ❌
4. **Neighbor Control:**
   - Explicit `neighbor` commands are used (good practice). ✅
5. **Redistribution:**
   - `redistribute connected` is configured, which may **expose internal networks** into RIP unnecessarily. CIS recommends minimizing redistribution to reduce attack surface. ❌
6. **Administrative Distance:**
   - Set to `150` (custom), which is higher than default (120). This is acceptable as long as it's intentional. ✅
7. **Access Control:**
   - No explicit **distribute-lists** or **route filtering** for RIP updates. CIS recommends filtering routes to avoid accepting unauthorized updates. ❌
8. **Other Security Observations:**
   - `no service password-encryption` is configured early in the file (later overridden by `service password-encryption`), which is inconsistent.
   - Console and AUX lines have **no login with privilege level 15**, which is a significant security risk. ❌

***Figure 4.6:*** *Specific Prompt Output Example*

sampled, and GPT's references recorded. CIS had been referenced in all 12 configurations and Cisco in 11 of them which were the most commonly referenced, followed by NIST with 8 references, real-world application sources with 4 and DIS STIG guidelines referenced 2 times. This demonstrates that although the prompt specifically instructed GPT to use the CIS Benchmarks, it continued to integrate information from a range of external sources. While Cisco and CIS remained dominant, the presence of other frameworks such as NIST and DISA STIG suggests that the model blended attaches standards with embedded training data, which may partly explain its inconsistent application of certain rules.

| Reference Source | Count |
|---|---|
| **CIS References** | **12/12** |
| **Cisco References** | **11/12** |
| **NIST References** | **4/12** |
| **Real-world applications Referenced** | **4/12** |
| **DISA STIG References** | **2/12** |

***Table 4.12:*** *Most common references used by Specific Prompt*

## 4.4 Cross-Reference Analysis with Other Studies

The findings of this study align closely with those reported by Sare and Debono [22], particularly in the effect that benchmark guidance has on LLM performance. In this research, the Broad prompt produced limited accuracy, with an overall PP Score of 28% for misconfiguration detection, whereas accuracy increased substantially when benchmark excerpts were introduced in the Specific case, reaching 61%. A similar trend was documented by Sare and Debono, who reported that

"GPT-4's response accuracy rate is 75%. Compared to GPT-4's performance from their Recorded results, without CIS benchmark document provided, which has an accuracy rate of 40.8%, this shows an improvement of 34.2%" [22]. Her work also demonstrated that baseline zero-shot prompting without CIS reference produced accuracy rated of only 40.8% for GPT-4 and 26.3% for GPT-3.5, highlighting the same challenge observed in this study, where unguided prompts struggled to consistently detect benchmark-aligned errors. Furthermore, Sare and Debono observed that when prompts explicitly referenced CIS benchmarks, "There is a 10% improvement in the 'Cross-reference with CIS Benchmarks' category" [22], which is consistent with the improvements noted here when using benchmark-guided prompts. Together, these parallels reinforce that while LLMs demonstrate some capacity for detecting vulnerabilities under general prompts, their accuracy is markedly enhanced when grounded in explicit compliance standards.

In addition to Sare's study, the results reported by Cao et al. [14] further validate the central findings of this dissertation, the degree of guidance in prompt design directly governs the effectiveness of LLMs in fault detection. Cao et al. observed that when ChatGPT was tasked with identifying faulty programs using a minimal prompt template, it successfully detected only 27 out of 34 cases, However, when supplied with a more informative and structured prompt, performance improved dramatically, reaching 34 out of 34 detections. A similar pattern emerged in this dissertation's evaluation of Cisco IOS configurations. Under the unguided Broad prompt, GPT identified 34 out of 120 misconfigurations. Introduction explicit references to CIS Benchmarks in the Mid prompt improved this slightly to 39 out of 120. The best results were achieved under the Specific prompt, which provided authoritative benchmark excerpts and narrowed the scope to a single protocol, allowing the model to identify 73 out of 120 misconfigurations. It was also reported that a similar pattern regarding the effect of prompt design on LLM performance was shown. It is shown that a basic, minimally guided prompt led ChatGPT to miss a meaningful portion of true faults and sometimes emphasize non-critical issues, whereas providing clearer task intention and context substantially improved performance and shifted the model towards a more functionally relevant direction, these results are consistent with this dissertations results, Unguided broad prompts tended to produce shorter, list-like outputs and occasional false positives, while benchmark-guided, protocol-scoped prompting yielded more structured compliance reasoning and higher accuracy in identifying materially important configuration issues.

The variability and inconsistencies observed in our three test cases align closely with results reported by Sobania et al. [5] on automatic program repair. In our setting, the models identified 34/120 misconfigurations under the Broad test (86 not found), 39/120 under the Mid test (81 not found) and 73/120 under the Specific, CIS-guided test (47 not found). In their benchmark, ChatGPT solved 19/40 buggy programs at baseline, a level comparable to other LLMs tested which are CoCoNut (19/40) and slightly below Codex (21/40), while traditional APR baselines solved 7/40 under a stricter generalization check, indicating that large portions of true faults remained unsolved without additional guidance [5]. Taken together, the specifics from both studies point to the same grading methods, without precise, task-oriented guidance, LLMs yield unstable and incomplete results (ChatGPT 19/40 for Sobania et al. and ours 34/120 under Broad), whereas clearer objectives and structured constraints materially improve out comes (ours 73/120 under Specific), with Sobania et al. further showing that targeted follow-up hints can raise ChatGPT's solved set to 31/40 when additional problem information is supplied.

## 4.5 Overall Test Cases Assessment

Across all three test cases, a clear progression was observed in the model's ability to detect benchmark-aligned errors, with accuracy increasing as prompts became more specific and guidance was made more explicit. The Broad prompt identified 34 out of 120 misconfigurations (28%), the Mid prompt rose slightly to 39 out of 120 (33%), while the Specific prompt achieved 73 out of 120 (61%). This steady improvement demonstrated that prompt engineering plays a pivotal role in shaping the reasoning process of large language models, validating the central hypothesis of this study that the precision and structure of prompts directly influence compliance assessment performance. At the protocol level, results highlighted persistent patterns of strength and weakness. OSPF consistently achieved the highest scores across all prompt types, suggesting that the models possess stronger embedded knowledge and rule application capacity for this protocol compared to others. By contract, EIGRP and RIP were the weakest domains throughout, with frequent failures to detect critical commands. These discrepancies underscore the uneven distribution of protocol knowledge within the models and highlight the importance of scope-specific evaluation. Qualitative analysis showed that as the prompts progressed, the model's outputs shifted from short, vague and list-like responses to longer, compliance-oriented explanations and finally to structured checklist-like reasoning anchored in CIS standards. However, even when benchmark excerpts were explicitly attached in the Specific case, the models continued to reference external sources, including Cisco documentation, NIST guidelines and real-world practices. This behaviour indicated that while LLMs can be directed toward authoritative standards, they cannot be fully constrained to them, raising concerns for professional use cases where strict adherence is a must. Taken together, these findings demonstrate both the potential and the limitations of LLMs in network configuration auditing. Prompt engineering was shown to markedly improve accuracy, yet detection remained inconsistent and heavily protocol-dependent, with many critical requirements still overlooked. While the Specific prompt achieved the highest accuracy, its 61% detection rate remains insufficient for production environments where full compliance is non-negotiable. These outcomes confirm that LLMs cannot yet replace formal auditing tools, but they do provide evidence that with carefully constructed prompts and controlled datasets, they can be leveraged as support tools in compliance checking workflows.

# Chapter 5: Conclusions and Recommendations

## 5.1 Summary of Findings

This dissertation hypothesised that large language models (LLMs) can improve the accuracy and efficiency of network configuration verification by detecting vulnerabilities and misconfiguration more effectively than traditional rule-based methods. The underlying assumption was that, with carefully engineered prompts and a structured evaluation framework, thesis models could deliver consistent compliance checks against recognised security benchmarks, such as the CIS standards, while also capturing more practical errors like typographical mistakes. The findings of this research partially confirmed this hypothesis. While the models demonstrated an ability to detect security misconfigurations and compliance failures, their performance was highly dependent on the quality and specificity of the prompt. Broad and unguided instructions led to weak and inconsistent results, whereas prompt that explicitly referenced compliance frameworks or provided benchmark excerpts yielded stronger outputs. Even so, the results showed that although LLMs have potential in this domain, their inconsistencies prevent them from being considered production-ready tools. The study systematically assessed benchmark aligned misconfigurations and Mistype errors across four major protocol domains: AAA, EIGRP, OSPF and RIP. The results showed a clear trend, the more structured and specific the prompt, the more effective the model was at detecting errors. The strongest outcomes were achieved with protocol-specific prompts supported by benchmark excerpts, which enabled the models to provide detailed and compliance-oriented reasoning. However, the models frequently overlooked certain critical requirements, and in some cases, they relied on external references not provided in the prompt, demonstrating the interpretive variability of LLM reasoning. The approach taken in this research, combining AI-generated configurations which were validated in GNS3, systematic error injection and manual inspection of outputs, was central to uncovering these findings. By deliberately structuring the dataset and controlling the evaluation proves, it was possible to isolate prompt specificity as the primary variable influencing accuracy. This approach not only highlighted the practical limitations of current LLMs but also revealed qualitative patterns, such as their tendency to hallucinate issues or misinterpret valid commands under unguided conditions. To support these evaluations, a layered framework was developed, progressing from broad to mid-level and then to specific prompt types. This framework was designed to measure reasoning performance across increasing levels of guidance, while using a transparent metric, the PP Score, to provide consistency in accuracy assessment. The framework proved effective in identifying both strengths and limitations, demonstrating that while LLMs can achieve meaningful detection rates under guided conditions, they remain prone to inconsistency when applied without structure, this layered design ultimately provided the evidence needed to critically evaluate the hypothesis and show where LLMs stand today in terms of readiness for network auditing and compliance verification.

## 5.2 Dissertation Limitations

This dissertation was subject to several limitations which shape the scope and interpretation of its findings. The first limitation relates to the coverage of security domains. Only four protocols were examined meaning that many other areas of the CIS Benchmarks, such as SNMP, logging or access control lists were not included in the evaluation. As such, the study does not claim to provide a comprehensive assessment of LLM capability across the full spectrum of Cisco IOS security requirements. A second limitation is the synthetic nature of the dataset. All router configuration were generated using LLM-based prompts rather than being collected from real-world production environments. While these configurations were validated for operational correctness in GNS3, they inevitably lack the unpredictability, legacy practices and context-specific design choices that characterise enterprise networks. This limitation was caused by the absence of openly available Cisco IOS 15 datasets, particularly for the C7200 router, which necessitated the creation of an artificial dataset tailored to this research. The third limitation concerns dataset size. The study employed 80 configurations in total, which is sufficient for controlled experimentation and proof-of-concept testing but remains small in scale compared to what would be required for a full benchmarking suite. This limited sample size means that results should be interpreted as indicative rather than exhaustive, capturing patterns in model behaviour without representing all possible network misconfigurations or error types. In conclusion, these limitations reflected the boundaries within which the research was conducted. Importantly, LLM technology is evolving rapidly, with frequent changes in model behaviour and capability. Findings presented here are therefore time-bound and future work may yield different outcomes as models improve, datasets expand and broader CIS domains are incorporated into evaluation frameworks.

## 5.3 Recommendations for Future Research

The findings of this dissertation highlight both the promise and the current limitations of using Large Language Models (LLMs) for network configuration auditing. While the evaluation demonstrated that structured prompt engineering can guide models toward improved compliance assessment, it also revealed inconsistencies that prevent immediate deployment in production environments. Based on these insights, several recommendations for future research are proposed to extend and refine this line of investigation. First, future work should broaden the scope of evaluation to include additional CIS Benchmark domains beyond AAA, EIGRP, OSPF and RIP. This study focused on four core protocols area due to time and dataset constraints, yet the CIS Benchmarks cover a wide range of services and security requirements. Expanding coverage to domains such as SNMP, NTP and IPv6 would provide a more comprehensive assessment of model performance across the full breadth of enterprise-grade configurations. Doing so would not only test the scalability of the evaluation framework but also generate deeper insights into whether LLMs generalise effectively across diverse security standards. Second, dataset expansion is critical. The present research relied on 80 synthetic configurations generated through carefully engineered prompts, which served as a valid proof-of-concept but did not capture the full variability of operational networks. Future studies should aim to incorporate a larger dataset and, where feasible, include anonymised real-world configurations. This would help validate whether the patterns observed here, such as the tendency to detect surface-level issues more reliably than protocol-specific misconfigurations, remain consistent when tested on authentic, complex environments. Third, refinement of the evaluation methodology is recommended. Manual inspection was the most effective approach in this study, ensuring that outputs were carefully verified against CIS requirements. However, as models evolve, incorporating automated scoring pipelines could improve scalability and reduce evaluator subjectivity. In particular, future research should consider multi-run variance testing, since repeated outputs may differ in subtle but important ways. Tracking this variability would provide a clearer picture of model consistency and reliability across multiple attempts. Fourth, prompt engineering strategies should continue to be refined and tested under different levels of specificity. This dissertation demonstrated that increasing guidance significantly improved detection rates, yet models still drew on external references even when instructed to rely solely on CIS Benchmarks. Future research should investigate mechanisms to enforce stricter alignment with provided reference materials, potentially through fine-tuning approaches. Additionally, exploring how interactive prompts or multi-step dialogues influence accuracy could offer valuable insights for real-world applications, where iterative auditing is common practice. Finally, while this dissertation confirmed that LLMs show potential as auditing assistants, it also made clear that they are not yet reliable replacements for human auditors. The field of AI is advancing rapidly, and models are continually updated with larger training sets, longer memory windows and improved reasoning capabilities. Future research should revisit the evaluation framework presented here at regular intervals, testing new models against the same CIS-aligned datasets to track progress over time. In conclusion, this dissertation has provided valuable insight into how prompt engineering and dataset design can be leveraged to assess LLM performance in security compliance auditing. The recommendations outlined above point toward a path for strengthening both the datasets and methodologies used in future studies. By expanding protocol coverage, incorporating real-world data, adoption scalable evaluation methods and refining prompt engineering strategies, subsequent research can build on this work to move closer to integrating LLMs into professional compliance workflows.

# List of References

[1] [Online]. Available: https://www.cisecurity.org/cis-benchmarks

[2] H. Li and L. Shan, "LLM-based Vulnerability Detection," *LLM-based Vulnerability Detection*, pp. 1–4, 12 2023. [Online]. Available: https://doi.org/10.1109/hccs59561.2023.10452613

[3] K. Pranav, "AI, ML, and large language models in cybersecurity," *International Research Journal of Modernization in Engineering Technology and Science*, 3 2024. [Online]. Available: https://doi.org/10.56726/irjmets49546

[4] D. Morrison, "The Convergence of AI and Cybersecurity: An Examination of ChatGPT's Role in Penetration Testing and its Ethical and Legal Implications," 2023. [Online]. Available: https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/3076387

[5] D. Sobania, M. Briesch, C. Hanna, and J. Petke, "An Analysis of the Automatic Bug Fixing Performance of ChatGPT," *IEEE/ACM International Workshop on Automated Program Repair*, 5 2023. [Online]. Available: https://doi.org/10.1109/apr59189.2023.00012

[6] N. Shenoy and A. V. Mbaziira, "An extended review: LLM Prompt Engineering in Cyber Defense," *2021 International Conference on Electrical, Computer and Energy Technologies (ICECET)*, pp. 1–6, 7 2024. [Online]. Available: https://doi.org/10.1109/icecet61485.2024.10698605

[7] P. Sharma and B. Dash, “Impact of big data analytics and chatgpt on cybersecurity,” 03 2023.

[8] M. I. Khan, A. Arif, A. Raza, and A. Khan, “The most recent advances and uses of ai in cybersecurity,” 04 2025.

[9] K. Okutu and H. Yumetoshi, “Explainability of Large Language Models (LLMs) in Providing Cybersecurity Advice,” 6 2024. [Online]. Available: https://doi.org/10.31219/osf.io/q84rk

[10] F. Wang, “Using Large Language Models to Mitigate Ransomware Threats,” 11 2023. [Online]. Available: https://doi.org/10.31219/osf.io/mzsnh

[11] Y. Huang, J. Xu, Z. Jiang, J. Lai, Z. Li, Y. Yao, T. Chen, L. Yang, Z. Xin, and X. Ma, “Advancing Transformer Architecture in Long-Context Large Language Models: A Comprehensive survey,” *arXiv (Cornell University)*, 1 2023. [Online]. Available: https://arxiv.org/abs/2311.12351

[12] D. De-Fitero-Dominguez, E. Garcia-Lopez, A. Garcia-Cabot, and J.-J. Martinez-Herraiz, “Enhanced automated code vulnerability repair using large language models,” *Engineering Applications of Artificial Intelligence*, vol. 138, p. 109291, 9 2024. [Online]. Available: https://doi.org/10.1016/j.engappai.2024.109291

[13] C. Zhang, H. Liu, J. Zeng, K. Yang, Y. Li, and H. Li, “Prompt-Enhanced software vulnerability detection using ChatGPT,” *arXiv (Cornell University)*, 1 2023. [Online]. Available: https://arxiv.org/abs/2308.12697

[14] J. Cao, M. Li, M. Wen, and S.-C. Cheung, “A study on Prompt Design, Advantages and Limitations of ChatGPT for Deep Learning Program Repair,” *arXiv (Cornell University)*, 1 2023. [Online]. Available: https://arxiv.org/abs/2304.08191

[15] R. Afsharmazayejani, M. M. Shahmiri, P. Link, H. Pearce, and B. Tan, “Toward Hardware Security Benchmarking of LLMs,” pp. 1–7, 6 2024. [Online]. Available: https://doi.org/10.1109/lad62341.2024.10691745

[16] A. P. H and G. Sujatha, “System Hardening using CIS Benchmarks,” *2022 International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI)*, pp. 1–6, 5 2024. [Online]. Available: https://doi.org/10.1109/accai61061.2024.10602274

[17] L. Long, R. Wang, R. Xiao, J. Zhao, X. Ding, G. Chen, and H. Wang, “On LLMS-Driven Synthetic Data Generation, Curation, and Evaluation: a survey,” *arXiv (Cornell University)*, 6 2024. [Online]. Available: https://arxiv.org/abs/2406.15126

[18] “Software Vulnerability and Functionality Assessment using LLMs,” 4 2024. [Online]. Available: https://ieeexplore.ieee.org/document/10647161

[19] R. Mondal, A. Tang, R. Beckett, T. Millstein, and G. Varghese, “What do LLMs need to Synthesize Correct Router Configurations?” *arXiv (Cornell University)*, 1 2023. [Online]. Available: https://arxiv.org/abs/2307.04945

[20] Y. Abdullin, D. Molla-Aliod, B. Ofoghi, J. Yearwood, and Q. Li, “Synthetic Dialogue Dataset Generation using LLM Agents,” *arXiv (Cornell University)*, 1 2024. [Online]. Available: https://arxiv.org/abs/2401.17461

[21] I. Martin, J. A. Hernandez, and O. G. De Dios, “Netgen: A Fast and Scalable Tool for the Generation and Labeling of Networking Datasets,” pp. 1–4, 7 2019. [Online]. Available: https://doi.org/10.1109/icton.2019.8840020

[22] A. Sare and D. Debono, “The Dual-Edged Sword: The Impact of Large Language Models in Network Infrastructure Security,” pp. 125–132, 1 2025. [Online]. Available: https://doi.org/10.5220/0013090700003899

# Chapter A: Introduction of Appendix

The appendices included in this dissertation provide supporting material that complements the main body of the research. These materials include the full prompts used for dataset generation, lists of benchmark-aligned commands used for error injection and supplementary configuration templates. They are presented to ensure transparency, reproducibility and clarity in the methodology. While the core dissertation discusses the structure and rationale of these elements, the appendices provide the complete reference texts and technical details that would otherwise be too extensive to include in the main chapters.

# Chapter B: IOS Template

```
!
!
!
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname R1
!
ip cef
no ip domain-lookup
no ip icmp rate-limit unreachable
ip tcp synwait 5
no cdp log mismatch duplex
!
line con 0
exec-timeout 0 0
logging synchronous
privilege level 15
no login
line aux 0
exec-timeout 0 0
logging synchronous
privilege level 15
no login
!
!
end
```

# Chapter C: Core Commands

```
"aaa new-model",
"aaa authentication login LOGIN-LIST group tacacs+ local",
"aaa authentication enable default group tacacs+ enable",
"aaa accounting connection CONN-ACC start-stop group tacacs+",
"aaa authorization exec EXEC-LIST group tacacs+",
"aaa accounting exec EXEC-ACC start-stop group tacacs+",
"aaa accounting network NET-ACC start-stop group tacacs+",
"aaa accounting system default start-stop group tacacs+",
"aaa accounting vrrs default start-stop group tacacs+",
"aaa accounting delay-start",
"Assign the TACACS+ server host IP to any address in the 10.0.0.0/24
subnet that has an interface associated with it.",
"Configure an 11-character randomly generated alphanumeric secret for
the TACACS+ server.",
"service password-encryption",
"Apply exec-timeout 10 0 to line vty 0 4",
"Apply transport input ssh to line vty 0 4",
"Apply access-class 10 in to line vty 0 4",
"Apply access-list 10 permit with an ip range 10.0.0.0 0.0.0.255"
```

# Chapter D: Extension block's

AAA:
"Configure a login banner with the text Ẅelcome to my C7200 UNAUTHORIZED ACCESS IS PROHIBITED. This device is monitored.̈.",
"Configure a failed-login banner with the text N̈ice try.̈.",
"Set the login timeout to any value between 1 and 10000 seconds.",
"Configure the RADIUS source interface to use an existing router interface.",
"AAA accounting for connection events with the method list CONN-ACC.",
"Configure AAA accounting for EXEC shell sessions with EXEC-ACC.",
"Configure AAA accounting for network services with NET-ACC.",
"Configure AAA accounting for system events with the default method list.",
"Under 'line con 0', configure 'login authentication LOGIN-LIST'.",
"Under 'line aux 0' configure 'login authentication LOGIN-LIST",
"Under 'line vty 0 4' configure 'login authentication EXEC-LIST",
"Apply aaa authorization config-commands",
"aaa authorization reverse-access default group tacacs+",
"aaa authorization commands 15 default group tacacs+ if-authenticated",
"aaa authentication login LOCAL-CASE local-case"

EIGRP:
"Under some of the EIGRP-enabled interface, configure: 'ip hello-interval eigrp ¡AS¿ ¡TIME¿', where TIME is a random number from 1 to 60000. This must not be placed under 'router eigrp', only under interface stanzas.",
"Enable log-neighbor-warnings with a seconds range randomly chosen from 1 to 65535", "Enable neighbor changes logging",
"Under each EIGRP-enabled interface, configure: 'ip hold-time eigrp ¡AS¿ ¡TIME¿', where TIME is a random number from 1 to 60000. This must not be placed under 'router eigrp', only under interface stanzas.",
"Under the EIGRP autonomous system redistribute OSPF with an area of your choice (from 0 to 4294967295) i.e redistribute ospf 5",
"Set a distance eigrp random number from 1 to 255 for internal and a random number from 1 to 255 for external. Do not use values above 255 — they are invalid and will break the configuration.",
"On one randomly chosen EIGRP-enabled interface, under the interface stanza, add: ip bandwidth-percent eigrp with a random number from 10 to 100",
"Set a passive interface",

”Configure traffic share across-interfaces on the EIGRP AS”,
”Configure EIGRP variance with a random number in the range of 1 to 128”,
”Configure EIGRP maximum-paths to a random number in the range of 1 to 32”,
”Configure EIGRP stub randomly choosing between the options”,
RIP:
”Under router rip, set version to 2.”,
”Under router rip, enable no auto-summary.”,
”Under router rip, enable passive-interface on a random set of interfaces.”,
”Under router rip, redistribute connected routes.”,
”Under router rip, redistribute static routes.”,
”Under router rip, redistribute ospf 1 metric 2.”,
”Under router rip, redistribute eigrp 100 metric 2.”,
”Under router rip, apply offset-list IN or OUT with a random hop count between 1 and 16.”,
”Set RIP timers basic with random values: update (5–60s), invalid (15–180s), holddown (15–180s), flush (30–240s).”,
”Apply distribute-list IN or OUT using a random ACL number.”,
”Enable RIP on a random subset of interfaces only.”,
”Apply authentication on RIP-enabled interfaces using key-chain rip-auth.”,
”On some interfaces, apply ip rip receive version 2.”,
”On some interfaces, apply ip rip send version 2.”,
”On some interfaces, apply ip rip authentication mode md5.”,
”On some interfaces, apply ip rip authentication key-chain rip-auth.”,
”Configure neighbor statements under router rip with fake neighbors from the same subnet.”,
”Apply maximum-paths ¡N¿ under router rip, where N is a random number between 1 and 6.”,
”Under router rip, use distance ¡value¿ where value is between 1 and 255.”,
”Under router rip, filter routes with route-map applied via distribute-list.”

OSPF:
”Under each ospf interface, set the ip ospf retransmit-interval to a random value in the range of 1 and 65535 seconds.”,
”Under each ospf chosen interface, set the ip ospf cost to a random value in the range of 1 and 65535.”,
”Under every ospf chosen interface, set the ip ospf priority to a number between 1 and 255.”,

```
"Under every ospf chosen interface, set the ip ospf hello-interval to a random number in the range of 1 to 60000.",
"Under every ospf chosen interface, set the ip ospf dead-interval to a random number in the range of 1 to 60000.",
"Under router ospf, if router has no Area 0 interface, add a virtual-link to a fake neighbor in Area 1.",
"Under router ospf, set 1 or 2 interfaces which have ospf enabled to passive-interface.",
"Under router ospf, set auto-cost reference-bandwidth to a random number within 1 to 4000000.",
"Under router ospf, redistribute (rip or eigrp) subnets.",
"Under ospf chosen interfaces, set ip ospf lls.",
"Under router ospf, add timers throttle spf X Y Z for X a random number within 1-60000, for Y a random number within 1-60000 for Z a random number within 1-60000",
"Under router ospf, add ispf.",
"Under router ospf, add default-information originate always.",
"Under router ospf, add area (x) nssa translate type7 (y), where x is the chosen area number and y is either always or suppress-fa.",
"Make some of the area's stubs or totally-stub under the same router ospf 1 configuration"
```

# Chapter E: Error Introduction commands

```
AAA:
    r'^aaa new-model',
    r'^aaa authentication login LOGIN-LIST group tacacs\+
       local$',
    r'^aaa authentication enable .*',
    r'^aaa authentication dot1x .*',
    r'^aaa authentication ppp .*',
    r'^aaa authentication arap .*',
    r'^aaa authentication attempts max-fail .*',
    r'^aaa authorization exec .*',
    r'^aaa authorization config-commands.*',
    r'^aaa authorization network .*',
    r'^aaa authorization reverse-access .*',
    r'^aaa accounting exec EXEC-ACC start-stop group
       tacacs\+$',
    r'^aaa accounting commands 15 .*',
    r'^aaa accounting connection .*',
    r'^aaa accounting network .*',
    r'^aaa accounting system .*',
    r'^aaa accounting vrrs .*',
    r'^aaa accounting delay-start',
    r'^aaa session-id common',
    r'^username .+ secret .+',
    r'^enable secret .+',
    r'^service password-encryption',
    r'^banner (exec|login|motd) [\s\S]+?\^C',
    r'^snmp-server community .+',
    r'^no snmp-server',
    r'^snmp-server host .+',
    r'^snmp-server enable traps snmp',
    r'^snmp-server group .+ v3 priv',
    r'^snmp-server user .+ v3 auth .+ priv aes 128 .+',
    r'^line con 0',
    r'^line tty \d+(?: \d+)?',
    r'^line aux 0',
    r'^line vty \d+(?: \d+)?'


EIGRP:
    r'^router eigrp \d+',
    r'^address-family ipv4 autonomous-system \d+',
    r'^af-interface .*',
    r'^authentication mode eigrp \d+ md5',
    r'^authentication key-chain eigrp \d+ \S+',
    r'^exit-af-interface',
    r'^exit-address-family',
    r'^key chain \S+',
    r'^key \d+',
    r'^key-string \S+',
    r'^ip authentication mode eigrp \d+ md5',
    r'^ip authentication key-chain eigrp \d+ \S+',
    r'^passive-interface \S+'
OSPF:

    r'^router ospf \d+$',
    r'^area \d+ authentication message-digest$',
    r'^ip ospf message-digest-key \d+ md5 .+$'

RIP:
    r'router rip',
    r'version 2',
    r'no auto-summary',
    r'ip rip authentication mode md5',
    r'ip rip authentication key-chain \S+',
    r'key chain \S+',
    r'key \d+',
    r'key-string \S+',
    r'passive-interface \S+',
    r'redistribute \S+',
    r'maximum-paths \d+',
    r'distance \d+',
    r'offset-list \d+ \S+'
```