Table of Contents

[Chapter 3: Research Methodology 3](#_Toc206865271)

[3.1 Introduction 3](#_Toc206865272)

[3.2 Hypothesis and Research Pipeline 4](#_Toc206865273)

[3.2.2 Research Pipeline 4](#_Toc206865274)

[3.3 Dataset Design and Preparation 5](#_Toc206865275)

[3.3.1 Dataset Generation 6](#_Toc206865276)

[3.3.2 Protocol Coverage and Scope 8](#_Toc206865277)

[3.3.3 Error Design 8](#_Toc206865278)

[3.3.4 Validation Against CIS Benchmarks 9](#_Toc206865279)

[3.3.5 Dataset Realism 9](#_Toc206865280)

[3.3.6 Limitations and Contribution 9](#_Toc206865281)

[3.4 Test Case Design 10](#_Toc206865282)

[3.4.1 Broad Assessment 10](#_Toc206865283)

[3.4.2 Mid-Level Assessment 10](#_Toc206865284)

[3.4.3 Specific, CIS-Guided Assessment 11](#_Toc206865285)

[3.5 Prompt Construction 11](#_Toc206865286)

[3.5.1 Broad Prompt 11](#_Toc206865287)

[3.5.2 Mid Prompt 12](#_Toc206865288)

[3.5.3 Specific Prompt 13](#_Toc206865289)

[3.5.4 Standardisation and Reproducibility 14](#_Toc206865290)

[3.6 Evaluation Procedure 15](#_Toc206865291)

[3.6.1 Execution of Test Cases 15](#_Toc206865292)

[3.6.2 Recording and Categorisation of Results 16](#_Toc206865293)

[3.6.4 Accuracy Focus 16](#_Toc206865294)

[3.6.5 Reproducibility Measures 17](#_Toc206865295)

[3.6.6 Limitations of the Evaluation Procedure 17](#_Toc206865296)

[3.7 Analysis Methods 18](#_Toc206865297)

[3.7.1 Quantitative Analysis 18](#_Toc206865298)

[3.7.2 Comparative Analysis 19](#_Toc206865299)

[3.7.3 Interpretation of Results 19](#_Toc206865300)

[References 20](#_Toc206865301)

[Appendix 20](#_Toc206865302)

[Appendix A 20](#_Toc206865303)

[Appendix B 21](#_Toc206865304)

# Chapter 3: Research Methodology

## 3.1 Introduction

This chapter discusses the methodology behind three proposed test cases designed to evaluate the compliance of Large Language Models (LLMs) specifically GPT-4o and GPT-5. These 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, 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, 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

This study is based on the hypothesis that large language models (LLMs) can improve the accuracy and efficiency of network configuration verification by detecting vulnerabilities and misconfigurations more effectively than traditional rule-based methods.

From this hypothesis, research questions were made to guide 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?

### 3.2.2 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

1. Phase 02: Formulation of Prompts

A series of prompts were developed for generating configurations and for evaluating them under different conditions. Prompts were designed for each protocol domain (AAA, EIGRP, OSPF, RIP), for merging multi-protocol configurations, for error introduction and for verification.

1. Phase 03: Dataset creation

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.

1. Phase 04: Analysis of Collected Data

Model responses were collected for each test case and assessed against the defined criteria and metrics.

1. Phase 05: Evaluation of Results

Finally, the outcomes were evaluated at two levels:

Criteria-based evaluation, determining whether CIS—aligned misconfigurations were correctly identified

Metric-based evaluation, quantifying accuracy across test cases and protocols

To achieve the objective, the steps outlined in Figure 1 were followed.

A diagram of a company

AI-generated content may be incorrect.

Figure 1: Research Pipeline

## 3.3 Dataset Design and Preparation

A custom dataset of Cisco IOS configurations was created for the purpose of evaluation 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 datasets exist for Cisco platforms such as the C7200 router, it was necessary to generate the dataset specifically for this dissertation. While originally designed for this study, the dataset has potential for reuse and extension in future research.

### 3.3.1 Dataset Generation

The dataset was generated using OpenAI’s API with carefully engineered prompts. These prompts were designed to mimic the work of a network architect constructing realistic Cisco IOS configurations. 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.

A shortened example of a representative OSPF prompt is shown below:

|  |
| --- |
| 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  Choose a random number from 1 to 6 then create that amount of OSPF areas on the chosen router  Randomly select all OSPF area IDs (from 0 to 4294967295) without reusing any.  Port Interfaces must include the advertised network to make it realistic.  Each block MUST start with for example "hostname R1" "hostname R2" etc.  For each router, assign it a unique hostname (e.g. R1, R2…).  Use only subnets in the 192.168.\*.\*/24 space for VLANs, Make sure to choose a random number from 1-254 to replace the "\*" in the ip's  For each router, assign it \*\*unique\*\* IP addresses for each link and OSPF interface  If the context lacks any required command, leave that section blank and write UNKNOWN.  \*\*Configure OSPF using half under the network statements under the router ospf 1 configuration mode. while the other half under interfaces. |

The full generation prompt, including the extended instruction block, is provided in Appendix A for reference.

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

### 3.3.2 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. Each domain contained 20 configurations, giving a balanced distribution across protocol types. The decision to focus on these four areas was pragmatic, 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. Extending the dataset to cover additional protocols such as BGP, SNMP or IPSec would have increased its variety, but this was beyond the scope and time constraints of this dissertation.

### 3.3.3 Error Design

Within each protocol domain, configurations were subdivided into four categories:

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

The misconfigurations were created programmatically using Python scripts that systematically removed benchmark-mandated commands. Each protocol was associated with a curated list of rules expressed as regular expressions (see Appendix B), ensuring that every removed command directly corresponded to a CIS requirement. For example, in the AAA domain this included lines such as “aaa new-model”, “enable secret” or “service password-encryption”, in EIGRP, commands such as “authentication mode md5” or “passive-interface” were targeted, while in OSPF and RIP, critical authentication and versioning commands were selectively removed. By varying the number of commands removed, configurations with different levels of non-compliance were produced.

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).

This dual approach ensured that the dataset tested both benchmark-related compliance failures and more practical day-to-day issues faced by network engineers.

### 3.3.4 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

### 3.3.5 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.

### 3.3.6 Limitations and Contribution

The dataset is subject to several limitations. First, it covers only four protocol domains, leaving other areas of the CIS Benchmark unexplored. Second, the configurations are synthetic and AI-Generated rather than drawn from real production environments. While they were tested in GNS3 for validity, they do not capture the full variability of operational networks. Finally, the dataset is limited in size, consisting of 80 configurations, which, while sufficient for proof-of-concept evaluation, does not constitute an exhaustive benchmark.

Despite these limitations, the dataset represents a meaningful contribution to the field. No comparable open-source Cisco IOS configuration dataset exists, particularly for C7200 devices. By combining benchmark-aligned misconfigurations with realistic, multi-protocol enterprise configurations, this dataset offers a foundation for future research in automated compliance checking, network misconfiguration detection, and the application of large language models in cybersecurity contexts.

## 3.4 Test Case Design

To evaluate the performance of GPT-4o and GPT-5 in identifying misconfigurations and assessing compliance with industry standards, three test cases were designed. These test cases varied in the degree of guidance provided to the model and the presence or absence of CIS Benchmark references, enabling a comparison of general reasoning, implicit benchmark knowledge and explicit benchmark application.

### 3.4.1 Broad Assessment

The first test case adopted a broad, open-ended approach. The model was prompted as a network security analyst reviewing a router configuration for deployment in a mid-sized corporate network. The task was to determine whether the configuration was secure, and if not, to identify any security issues, misconfigurations, Mistype errors or best practice violations. No reference was made to CIS Benchmarks in this case. The purpose of this test was to measure the model’s general interpretative ability to detect errors in network configurations without external guidance, simulating a scenario where an analyst requires a high- level review

### 3.4.2 Mid-Level Assessment

The second test case introduced CIS Benchmarks into the prompt, instructing the model to assess the configuration according to these standards. However, no excerpts of the CIS documents were provided, requiring the model to rely on its internal knowledge and training data to recall relevant compliance requirements. This case tested whether GPT could correctly apply benchmark-driven reasoning when prompted, despite not being given the official text. It also assessed the extent to which the models had internalised knowledge of the CIS Benchmarks and could distinguish compliance-related issues from general misconfigurations.

### 3.4.3 Specific, CIS-Guided Assessment

The third test case was the most constrained, providing the model with both a narrow focus and explicit benchmark excerpts, in this case, the model was instructed to review only one protocol within the configuration (e.g. RIP, OSPF, AAA or EIGRP) and to assess compliance against the attached section of the CIS Benchmark specific to that protocol. The prompt emphasized that the assessment should be limited strictly to the given protocol. This test case examined the model’s ability to apply prescriptive compliance rules when explicitly provided, and to restrict their analysis to a defined scope. By doing so, it measured how effectively GPT could operate under conditions closely aligned with professional compliance audits.

Together, these three test cases provided a layered evaluation of Gpt-4o and GPT-5. The Broad case tested open-ended reasoning, The Mid case assessed the models internalised benchmark knowledge, and the Specific case evaluated benchmark-driven compliance checking within a constrained scope. This progression allowed the study to capture not only overall detection accuracy, but also the effects of prompt specificity, context and external guidance on the model’s performance.

### 3.5 Prompt Construction

Prompt construction was central to this study, as it determined how GPT-4o and GPT-5 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: Broad, Mid 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.

### 3.5.1 Broad Prompt

The Broad prompt was designed to simulate a general, high-level review without any reference to compliance frameworks. It framed the model as a network security analyst and instructed it to identify any potential misconfigurations, vulnerabilities, typographical errors, or violations of best practice. If no issues were present, the model was required to state explicitly that the configuration was secure.

Broad Prompt:

|  |
| --- |
| 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. |

The purpose of this test case was to establish a baseline for the model’s performance in an unguided scenario. It effectively measured the extent to which GPT could apply its generalised security reasoning to configurations without external standards. However, the Broad prompt revealed significant limitations, the models often failed to detect genuine CIS-related violations, while occasionally flagging non-issues as problems. This demonstrated that although LLMs possess general networking knowledge, their reliability in compliance auditing without guidance is low.

The Broad prompt was deliberately kept short and simple, as longer instructions often give LLMs too much room for misinterpretation. By using minimal wording, the prompt tested the model’s baseline reasoning capacity without introducing unnecessary complexity. However, in practice, GPT sometimes misclassified valid configuration elements as mistakes, for example, it frequently flagged the command “aaa accounting vrrs” as a typo, even though it was valid. This highlights a key limitation of unguided prompting, while the model is capable of recognising general misconfigurations, it is prone to introducing false findings.

### 3.5.2 Mid Prompt

The Mid prompt built on the Broad case by explicitly referencing the CIS benchmarks. However, no benchmark excerpts were provided, instead, the models were expected to draw on any internalised knowledge of the CIS standards gained during training. The intention was to test whether GPT had sufficient embedded knowledge of compliance frameworks to improve accuracy when prompted to use them.

Mid prompt:

|  |
| --- |
| 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. |

Compared to the Broad prompt, responses to the Mid prompt were better structured and more compliance-focused. GPT tended to identify more misconfigurations and frame its findings in a checklist-like style, even though I did not have direct access to the benchmark text. This suggests that referencing the CIS Benchmarks activated more structured reasoning patterns within the model. Either way, accuracy was still inconsistent, and the absence of explicit benchmark excerpts meant that GPT occasionally overgeneralised or applied rules incorrectly.

This prompt effectively tested whether GPT contained embedded knowledge of CIS rules from training. Its outputs tended to be more structured and checklist-like than in the Broad case, suggesting that the references to CIS triggered more compliance-focused reasoning. Nevertheless, without direct access to benchmark excerpts, the model often overgeneralised. For instance, in several cases it provided security recommendations that were best-practice improvements but not explicitly part of the CIS Benchmarks. This reinforced the need for the Specific prompt, where the model was provided with authoritative rules.

### 3.5.3 Specific Prompt

The Specific prompt represented the most constrained and benchmark-driven test case. Unlike the Broad and Mid prompts, this case narrowed the task to a single protocol (AAA, EIGRP, OSPF or RIP) and attached the full corresponding section of the CIS Benchmark as reference material. The model was required to assess compliance only for the specified protocol, providing a binary statement of compliance if secure, or a list of issues if violations were detected.

For this case, full protocol-specific sections of the CIS Benchmarks were attaches as separate files, ensuring that the model had access to exact standards for AAA, EIGRP, OSPF or RIP. Despite this, the model was not always consistent. For example, in OSPF configurations it frequently failed to flag the absence of “ip ospf message-digest-key md5” on interfaces, even though the attached benchmark explicitly mandated its presence. This demonstrated that while providing benchmark excerpts improved accuracy, the model did not always apply the rules correctly, highlighting the interpretive gap between human auditors and LLMs.

Specific Prompt:

|  |
| --- |
| 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. |

The attached excerpts consisted of the full sections of the CIS Benchmark relevant to the given protocol, ensuring that GPT was provided with the authoritative rules for evaluation. The prompt type effectively tested the model’s ability to apply explicit compliance standards rather than relying solely on internalised knowledge. It also provided the most realistic simulation of how an LLM might be used in a professional security audit, where official standards are available to guide the review.

### 3.5.4 Standardisation and Reproducibility

To maintain consistency, all three prompts were applied word-for-word across every configuration without modification. This eliminated variability in wording as a factor, ensuring that differences in model performance could be attributed solely to the level of specificity provided in the prompts. Early experiments with running the same prompt multiple times showed almost identical outputs, so for efficiency each configuration was evaluated only once per prompt.

Outputs were recorded manually rather than through automated pipelines. This approach was chosen deliberately, as the purpose of the study was to test the reasoning of LLMs rather than to allow the models to judge themselves. Manual logging ensured that each response could be inspected directly for accuracy, allowing for careful differentiation between correct detections, false positives and false negatives.

The evaluation criteria were intentionally simple. If the configuration was compliant, the model was required to output a statement such as “Yes, this configuration appears secure.” If it was not compliant, the model was expected to briefly list the issues. The binary structure facilitated consistent interpretation of results while reducing the influence of verbosity or stylistic variation in model responses

Manual inspection was chosen over automated result collection not only to verify correctness, but also to capture qualitative aspects of the outputs. By reading the responses in full, It was possible to observe how the model justified its findings, when it drew comparisons to CIS standards, and when it misinterpreted commands. This provided valuable insight into the reasoning process behind both correct and incorrect outputs, which would have been purely automated accuracy scoring. Early testing also showed that repeated runs of the same prompt produced almost identical results, so each configuration was evaluated once per prompt to ensure efficiency.

## 3.6 Evaluation Procedure

The evaluation procedure defined how the dataset, prompts and models were combined to assess the ability of GPT-4o and GPT-5 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.

### 3.6.1 Execution of Test Cases

Each of the 80 configurations in the dataset was tested once with all three prompt types (Broad, Mid, Specific). Initial testing was performed on GPT-4o, but partway through the evaluation model was discontinued, and GPT-5 was used to complete the final few runs such as the final few runs of the specific section and half of the Mid runs (GPT-4o is now available once again as a legacy model).This resulted in a mixed-model datasheet, with some variability in style and accuracy noted between the two. GPT-5, for instance, tended to produce longer, more elaborate explanations, whereas GPT-4o responses were more concise.

On average, each test run (entering the prompt, providing the configuration, and recording the result) took approximately 20 seconds per configuration for GPT-4o while taking slightly longer with approximately 45 seconds for GPT-5, making the evaluation process time-efficient despite being conducted manually. The full evaluation was carried out over multiple sessions spanning several days. Configurations were tested in a fixed order for consistency, beginning with AAA set, followed by EIGRP, OSPF then RIP. Within each domain, configurations were sequentially evaluated across the categories of single error, double error, triple error and Mistype error.

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 carried over between tests.

### 3.6.2 Recording and Categorisation of Results

All results were logged manually in an Excel spreadsheet that served as the central record of evaluation. Each prompt type (Broad, Mid, Specific) was separated into its own sheet, while all 80 configurations were kept together for ease of calculation. To ensure traceability, every GPT output was screenshotted and cross-referenced with the corresponding entry in the spreadsheet, allowing later verification if discrepancies arose.

For each run, results were categorised into four possible outcomes:

Yes: all injected errors were correctly identified.  
No: No errors were identified.  
1 Error/ 2 Errors: only 1 or 2 Errors were correctly identified

The spreadsheet also recorded the specific protocol domain (AAA, EIGRP, OSPF or RIP) and the error type (CIS Misconfiguration or Mistype error). This allowed the analysis to compare results both across protocols and between error categories.

### 3.6.4 Accuracy Focus

The evaluation focused exclusively on quantitative detection accuracy. Although both GPT-4o and GPT-5 frequently produced detailed explanations alongside their detections, these were not scored formally. Instead, the emphasis remained on whether the models correctly identified the required issues. Nevertheless, qualitative differences were noted during manual inspection, Broad prompts often produced free-form narrative-style responses, Mid prompts yielded structured, checklist-like outputs and Specific prompts produced more formal, benchmark-driven reports though occasionally the model over-relied on repeating CIS text rather than applying it.

### 3.6.5 Reproducibility Measures

Reproducibility was maintained through strict standardisation. All prompts were fixed word-for-word across all runs, ensuring that variability in results could not be attributed to changes in instruction. Configurations were input in a fixed sequence ( AAA > EIGRP > OSPF > RIP) and in consistent format (.txt attachments for Broad and Mid, copy-paste for Specific). After every test, the chat session was reset to eliminate the influence of conversational memory.

The Excel sheet was saved at the end of each protocol section, providing incremental backups throughout the evaluation. While the workflow could theoretically be replicated using the dataset, prompts and spreadsheet, some manual judgment calls were unavoidable. For example, in unclear cases, it had to be decided whether the models flagged issue aligned with the created CIS violation or represented a false positive.

### 3.6.6 Limitations of the Evaluation Procedure

Despite being systematic and controlled, the evaluation procedure carried certain limitations. First, the reliance on manual logging and judgment introduced a degree of subjectivity, particularly in cases where GPT flagged issues that were not explicitly benchmark violations but resembled best-practice improvements. Second, the switch from GPT-4o to GPT-5 mid-study introduced minor variability in style and accuracy that could not be perfectly normalised. Third, only single runs per configuration were performed, while early testing confirmed stability, the possibility of occasional non-deterministic outputs cannot be fully excluded. Finally, although screenshots ensured traceability, the reliance on manual logging limited scalability compared to automated methods.

## 3.7 Analysis Methods

The purpose of the analysis was to interpret the outputs generated by GPT-4o and GPT-5 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 models behaviour.

### 3.7.1 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 [14]. 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:

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

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

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

Mistype errors were measured only at the total dataset level.

### 3.7.2 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.

Visual comparison was conducted using bar charts for clarity and ease of interpretation. Each chart represented one of the three prompts, with the four protocols (AAA, EIGRP, OSPF and RIP) displayed along the x-axis and the PP scores on the y-axis. This allowed side-by-side comparison of protocol-level performance within a given prompt type, making it straightforward to see whether accuracy varied across domains. In addition, aggregated totals for misconfigurations and Mistype errors were also presented graphically. Bar charts were selected over other formats such as line graphs or stacked charts because they provide the clearest view of categorical comparisons.

### 3.7.3 Interpretation of Results

The central aim of the analysis was to evaluate how effectively large language models can detect security vulnerabilities and misconfigurations in network configurations. This study did not benchmark the models against existing tools or industry averages, but rather treated the evaluation as a standalone assessment of GPT performance in this task.

Interpretation focused on whether the models could correctly identify errors that represented real-world security risks, as defined by CIS Benchmarks, as well as whether they could detect common Mistype mistakes. Results were compares across the three prompt types to assess how levels of guidance influenced accuracy. The Broad prompt tested unguided reasoning, the Mid prompt measured the effect of referencing security standards without direct excerpts and the Specific prompt evaluated performance when authoritative benchmark text was provided.

In this way, improvements in PP score were interpreted as evidence that prompt specificity improves an LLM’s ability to perform security-focused configuration audits. Conversely, missed detections or false positives were taken as indicators of the limitations of current models in reliability performing such tasks without human oversight.

## References

[14] 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, Dec. 2024, doi: [10.1016/j.engappai.2024.109291](https://doi.org/10.1016/j.engappai.2024.109291).

## Appendix

### Appendix A

The dataset of Cisco IOS configurations used in this research was generated using OpenAI’s API with a structured prompt designed to mimic the work of a network architect. The prompt ensured that configurations reflected realistic enterprise-grade router deployments while incorporating randomisation to maximise diversity

The main prompt used to generate OSPF configurations is shown below

|  |
| --- |
| 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  Choose a random number from 1 to 6 then create that amount of OSPF areas on the chosen router  Randomly select all OSPF area IDs (from 0 to 4294967295) without reusing any.  Port Interfaces must include the advertised network to make it realistic.  Each block MUST start with for example "hostname R1" "hostname R2" etc.  For each router, assign it a unique hostname (e.g. R1, R2…).  Use only subnets in the 192.168.\*.\*/24 space for VLANs, Make sure to choose a random number from 1-254 to replace the "\*" in the ip's  For each router, assign it \*\*unique\*\* IP addresses for each link and OSPF interface  If the context lacks any required command, leave that section blank and write UNKNOWN.  \*\*Configure OSPF using half under the network statements under the router ospf 1 configuration mode. while the other half under interfaces.  {ext\_block}  === CONTEXT SNIPPETS ===  {context\_snippets}  === IOS TEMPLATE ===  {ios\_template}  === INSTRUCTIONS ===  Fill in or extend the attached C7200 Router IOS template above so that it configures:  - OSPF  - All networks chosen to be advertised must be  - 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, interface Serial2/4, interface Serial2/5, interface Serial2/6, interface Serial2/7 with the exception of subinterfaces like interface FastEthernet0/0.10  - Under router ospf, set the area authentication message-digest (use the same area number as for the OSPF area).  - For each interface that participates in OSPF, within its interface block add a line: ip ospf message-digest-key 1 md5 <KEY>. `<KEY>` must be a unique, randomly generated 8-character alphanumeric string per interface.  \*\*Output only\*\* the final, completed CLI configuration (no explanations), \*\*Create 10 router configurations\*\* |

The extended instruction block (ext\_block) was used to introduce further configuration diversity asfollows. Seven of these were randomly sampled for each generation run.

|  |
| --- |
| "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" |

### Appendix B

The following list defines the regular expressions used to programmatically identify AAA-related commands within the Cisco IOS configurations. During dataset generation, the error injection script randomly selected and removed one or more of these commands depending on the target error category (single error, double error or triple error). Each reg vex corresponds to a command mandated b y the CIS Cisco IOS Benchmarks for IOS 15. The removal of any listed command constitutes a deliberate security misconfiguration.

|  |
| --- |
| 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+)?' |