Table of Contents

[Chapter 3: Research Methodology 3](#_Toc207311754)

[3.1 Introduction 3](#_Toc207311755)

[3.2 Hypothesis and Research Pipeline 3](#_Toc207311756)

[3.2.1 Research Pipeline 4](#_Toc207311757)

[3.3 Dataset Design and Preparation 5](#_Toc207311758)

[3.3.1 Protocol Coverage and Scope 5](#_Toc207311759)

[3.3.2 Dataset Generation 6](#_Toc207311760)

[3.3.3 Prompt Structure 6](#_Toc207311761)

[3.3.4 Error Design 11](#_Toc207311762)

[3.3.5 Validation Against CIS Benchmarks 13](#_Toc207311763)

[3.3.6 Dataset Realism 13](#_Toc207311764)

[3.4 Prompt Construction 13](#_Toc207311765)

[3.4.1 Broad Assessment 13](#_Toc207311766)

[3.4.2 Mid-Level Assessment 14](#_Toc207311767)

[3.4.3 Specific, Cis-Guided Assessment 14](#_Toc207311768)

[3.4.4 Standardisation and Reproducibility 15](#_Toc207311769)

[3.5 Evaluation Procedure 16](#_Toc207311770)

[3.5.1 Execution of Test Cases 16](#_Toc207311771)

[3.6 Analysis Methods 16](#_Toc207311772)

[3.6.1 Quantitative Analysis 17](#_Toc207311773)

[3.6.2 Comparative Analysis 17](#_Toc207311774)

[References 18](#_Toc207311775)

[Appendix 19](#_Toc207311776)

[Appendix A 19](#_Toc207311777)

[Appendix B 19](#_Toc207311778)

[Appendix C 20](#_Toc207311779)

[Appendix D 21](#_Toc207311780)

# 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?

### A diagram of a company AI-generated content may be incorrect.3.2.1 Research Pipeline

Figure 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

1. 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.

1. 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.

1. Phase 04: Error introduction

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

1. Phase 05: Verification and Analysis

The results were evaluated at two levels:

* 1. Criteria-based evaluation was performed by determining whether CIS-aligned misconfigurations were correctly identified
  2. Metric-based evaluation was done by quantifying accuracy across test cases and protocols

1. Phase 06: Evaluation of Metrics

Results were evaluated and recorded accordingly 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 1 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 1: 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 2: 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 \"Welcome to my C7200 UNAUTHORIZED ACCESS IS PROHIBITED. This device is monitored.\".",      "Configure a failed-login banner with the text \"Nice 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: 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 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, interface 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 OPTIONAL.  - 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 explanations), \*\*Create 10 router configurations\*\* |

Figure 5: 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. "          "\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"          "\n"          "👷 Additional Instructions:\n"          "- All interface blocks must be merged together (no duplicates).\n"          "- Subinterfaces (e.g., FastEthernet0/0.100) behave like normal interfaces.\n"          "- Order the final config as follows: OSPF → EIGRP → RIP → AAA.\n"          "- If any protocol config is missing, include a placeholder: '<protocol> section: [UNKNOWN]'.\n"          "- Output ONLY the final merged configuration text, no explanation." |

Figure 6: 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 syntactic 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 7: 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” [13], as shown in Figure 8

|  |
| --- |
|  |

Figure 8: CIS AAA Example [13]

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.

### 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 9: 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 10: Mid-Level 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 11: Specific 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 [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:

1. Total dataset accuracy, expressed separately for misconfiguration and Mistype errors. Misconfigurations were benchmark-driven violations, while Mistype errors represented general robustness checks.
2. Per-protocol accuracy, covering AAA, EIGRP, OSPF and RIP. This enabled the analysis to highlight whether some protocols presented greater challenges for the models.
3. 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.

### 3.6.2 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.

## References

[13] A. P. H and G. Sujatha, ‘System Hardening using CIS Benchmarks’, in *2024 International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI)*, May 2024, pp. 1–6. doi: [10.1109/ACCAI61061.2024.10602274](https://doi.org/10.1109/ACCAI61061.2024.10602274).

[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

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 |

### Appendix B

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" |

### Appendix C

Extension block’s:

|  |
| --- |
| AAA:    "Configure a login banner with the text \"Welcome to my C7200 UNAUTHORIZED ACCESS IS PROHIBITED. This device is monitored.\".",      "Configure a failed-login banner with the text \"Nice 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" |

### Appendix D

Lists of commands used for error introduction:

|  |
| --- |
| 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+' |