Table of Contents

[Chapter 3: Research Methodology 3](#_Toc207124924)

[3.1 Introduction 3](#_Toc207124925)

[3.2 Hypothesis and Research Pipeline 4](#_Toc207124926)

[3.2.1 Research Pipeline 4](#_Toc207124927)

[3.3 Dataset Design and Preparation 5](#_Toc207124928)

[3.3.1 Protocol Coverage and Scope 6](#_Toc207124929)

[3.3.2 Dataset Generation 6](#_Toc207124930)

[3.3.3 Prompt Structure 7](#_Toc207124931)

[3.3.4 Error Design 10](#_Toc207124932)

[3.3.5 Validation Against CIS Benchmarks 11](#_Toc207124933)

[3.3.6 Dataset Realism 11](#_Toc207124934)

[3.3.7 Limitations and Contribution 11](#_Toc207124935)

[3.4 Test Case Design 12](#_Toc207124936)

[3.4.1 Broad Assessment 12](#_Toc207124937)

[3.4.2 Mid-Level Assessment 12](#_Toc207124938)

[3.4.3 Specific, CIS-Guided Assessment 12](#_Toc207124939)

[3.5 Prompt Construction 13](#_Toc207124940)

[3.5.1 Test prompts 13](#_Toc207124941)

[3.5.2 Mid and Specific Prompt 14](#_Toc207124942)

[3.5.3 Standardisation and Reproducibility 16](#_Toc207124943)

[3.6 Evaluation Procedure 16](#_Toc207124944)

[3.6.1 Execution of Test Cases 16](#_Toc207124945)

[3.7 Analysis Methods 17](#_Toc207124946)

[3.7.1 Quantitative Analysis 17](#_Toc207124947)

[3.7.2 Comparative Analysis 17](#_Toc207124948)

[References 18](#_Toc207124949)

[Appendix 18](#_Toc207124950)

[Appendix A 18](#_Toc207124951)

[Appendix B 19](#_Toc207124952)

[Appendix C 19](#_Toc207124953)

[Appendix D 20](#_Toc207124954)

[Appendix E 20](#_Toc207124955)

# Chapter 3: Research Methodology

## 3.1 Introduction

This chapter discusses the methodology behind three proposed test cases designed to evaluate Large Language Models (LLMs) specifically GPT-4o. 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?

### 3.2.1 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: 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

A diagram of a process

AI-generated content may be incorrect.To achieve the objective, the steps outlined in Figure 1 were followed.

Figure : Research Pipeline

## 3.3 Dataset Design and Preparation

A custom dataset of Cisco IOS 15 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 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.

### 3.3.2 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 15 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.

### 3.3.3 Prompt Structure

Introduction from prompt, this assigned the model a role and a defined task

|  |
| --- |
| You are a network architect.  \*\*Use ONLY the following context\*\* from the provided PDF manuals—do not rely on any other knowledge.  \*\*Create 50 router configurations\*\*  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\*\* |

Appendix A

This framing ensured that GPT approached the task as if it were designing an enterprise-grade router config, while grounding outputs in official Cisco documentation

Core commands, This was a set of mandatory commands required on every configuration to ensure baseline compliance with CIS-recommended security practices. For AAA, these included commands such as below

|  |
| --- |
| "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 B

Embedding these rules directly in the prompt guaranteed their inclusion, forming a reliable baseline for error injection later.

Extended block (ext\_block), A larger pool of optional protocol specific commands designed to increase diversity. At runtime, nine of these commands were sampled randomly using python. The commands are as shown below. This ensured that while all configurations followed a standard structure, no two were identical, better reflecting the variability of real-world deployments.

|  |
| --- |
| "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" |

Appendix C

Context snippets, these were extracts from Cisco manuals embedded using text-embedding-ada-002 to supply authoritative grounding. This step reduced the risk of GPT hallucinating syntax and ensured that generated configurations aligned with valid IOS 15 commands.

Final instructions, At the end of each prompt, strict rules were defined for how the configuration must be completed. These rules standardised the output format and forced uniqueness across configurations

|  |
| --- |
| 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\*\* |

Appendix A

Each prompt run produced 10 router configurations, with the overall dataset comprising 80 files.

After protocol specific configurations were generated, they were combined into composite multi-protocol files using a dedicated merging prompt. This ensures 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 below

|  |
| --- |
| "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" |

Appendix D

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

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, C), This ensured that every removed command directly corresponded to a CIS benchmark requirement, maintaining fidelity to recognised security standards.

For example, in the AAA domain, misconfigurations were created by removing critical lines such as

|  |
| --- |
| r'^aaa new-model',  r'^aaa authentication login LOGIN-LIST group tacacs\+ local$',  r'^aaa authorization exec .\*',  r'^aaa accounting exec EXEC-ACC start-stop group tacacs\+$',  r'^enable secret .+',  r'^service password-encryption',  r'^snmp-server community .+',  r'^line vty \d+(?: \d+)?' |

Appendix E

These deletions correspond to essential CIS requirements for enabling AAA services, protecting administrative access and securing SNMP

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 (The same as in Appendix E for example) to adjust.

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.

### 3.3.7 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 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. 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 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 Test prompts

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 and Specific Prompt

While the Broad prompt provided a baseline assessment of GPT’s unguided reasoning, two further prompts were designed to evaluate how referencing security standards influenced performance. These were the Mid prompt and the Specific prompt, which differed in the degree of guidance provided from the CIS Benchmarks

The Mid prompt explicitly referenced the CIS Benchmarks but did not supply any benchmark text. Instead, the models were expected to rely on their internalised knowledge of compliance frameworks learned during training. This prompt tested whether invoking the benchmarks by name would encourage GPT to apply more structured, standard driven reasoning. Responses under this condition were typically more checklist-like and compliance-focused than in the Broad case, the accuracy was inconsistent. At times the models overgeneralised, offering best-practice recommendations that were not strictly required by CIS, highlighting the limitations of relying solely on embedded knowledge.

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. |

The Specific prompt introduced the highest level of guidance by narrowing the evaluation to a single protocol and attaching the relevant excerpt of the CIS Benchmark. This tested the model’s ability to apply explicit rules directly rather than relying on inference. In practice, the availability of benchmark text improved both accuracy and structure of responses, but inconsistencies remained. For example, in OSPF configurations the model often failed to flag the absence of “ip ospf message-digest-key md5” on interfaces, even though the benchmark explicitly required it. Such cases illustrated the interpretive gap between humans and LLMs even when authoritative standards were provided.

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. |

Together, the Mid and Specific prompts formed a continuum of compliance-oriented evaluation. The Mid prompt tested GPTs latent knowledge of CIS, while the Specific prompt simulated a realistic professional audit scenario in which official standards are provided. Comparing these against the Broad baseline allowed this study to explore how varying levels of guidance shaped the model’s effectiveness at detecting security vulnerabilities.

### 3.5.3 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 produces near-identical outputs, so each configuration was evaluated once per prompt to balance accuracy with efficiency.

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.

## 3.6 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.

### 3.6.1 Execution of Test Cases

Each of the 80 configurations in the dataset were tested once with all three prompt types (Broad, Mid, Specific) and was performed on the model GPT-4o

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.\

## 3.7 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.

### 3.7.1 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.

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

[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 15 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 AAA 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.  \*\*Create 50 router configurations\*\*  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\*\*  === \*\*CORE AAA COMMANDS (Include on EVERY router):\*\* ===  {core\_block}  === EXTENSIONS (Include all of the following on EVERY router ===  {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:  - 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\*\* |

### 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" |

### 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" |

### Appendix D

|  |
| --- |
| "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." |

### Appendix E

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