Table of Contents

[Chapter 3: Research Methodology 2](#_Toc206616476)

[3.1 Introduction 2](#_Toc206616477)

[3.2 Hypothesis and Research Pipeline 3](#_Toc206616478)

[3.2.2 Research Pipeline 3](#_Toc206616479)

[3.3 Dataset Design and Preparation 4](#_Toc206616480)

[3.3.1 Dataset Generation 5](#_Toc206616481)

[3.3.2 Protocol Coverage and Scope 5](#_Toc206616482)

[3.3.3 Error Design 5](#_Toc206616483)

[3.3.4 Validation Against CIS Benchmarks 6](#_Toc206616484)

[3.3.5 Dataset Realism 6](#_Toc206616485)

[3.3.6 Limitations and Contribution 7](#_Toc206616486)

[3.4 Test Case Design 7](#_Toc206616487)

[3.4.1 Broad Assessment 7](#_Toc206616488)

[3.4.2 Mid-Level Assessment 8](#_Toc206616489)

[3.4.3 Specific, CIS-Guided Assessment 8](#_Toc206616490)

[3.5 Prompt Construction 8](#_Toc206616491)

[3.6 Evaluation Procedure 8](#_Toc206616492)

[3.7 Analysis Methods 8](#_Toc206616493)

# 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 typographical 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 : 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" |

The use of these 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

### 3.3.2 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 typographical 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 contract, the typographical 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 t he 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, typographical 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 ana 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 extend 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

### 3.6 Evaluation Procedure

### 3.7 Analysis Methods

### 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+)?' |