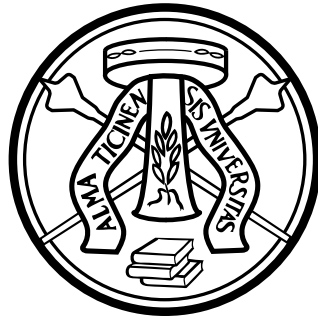




UNIVERSITÀ DI PAVIA

UNIVERSITY OF PAVIA
FACULTY OF ENGINEERING
DEPARTMENT OF ELECTRICAL, COMPUTER AND BIOMEDICAL
ENGINEERING

MASTER'S DEGREE IN COMPUTER ENGINEERING

MASTER THESIS

An Evidence-Grounded Framework for Traceable Cybersecurity Compliance Assessment

**Un Framework basato su Evidenze per Valutare e
Tracciare la Conformità in ambito Cybersecurity**

Candidate: Eylül Naz Özcan

Supervisor: Prof. Maria Carla Calzarossa

Co-supervisor: Dr. Konrad Wrona

Academic Year 2025/2026

Abstract

The increasing dependence of modern society on information systems has made cybersecurity compliance increasingly important for organizations that must demonstrate that their systems satisfy applicable security requirements. This assessment requires connecting high-level regulatory requirements with concrete evidence from the system being assessed. This is challenging since the information to evaluate compliance is distributed across system configurations, infrastructure definitions, runtime state, and other technical artifacts. In addition to producing a compliance decision, an assessment should preserve a clear connection between the decision and the evidence on which it is based.

This thesis work presents a methodology for evidence-grounded and traceable cybersecurity compliance assessment. The proposed methodology organizes the assessment into three main stages: security-control mapping, evidence grounding, and compliance evaluation. Regulatory requirements are mapped to standardized security controls, while relevant properties collected from the assessed system are represented as structured observations. The requirement, its security-control context, and the selected observations form a grounded assessment case, which is evaluated using either deterministic rules or a language-model-based evaluator. Evidence provenance is retained throughout the process to support traceability of the resulting decision.

The proposed methodology is assessed through a proof-of-concept implementation and a set of controlled experiments. Four experimental configurations are used to examine different combinations of expert and model-generated security-control mappings, manual and model-generated observation selections, and rule-based and language-model-based compliance evaluation. The results show that changes in security-relevant system properties can be reflected in the collected evidence and propagated to the resulting compliance assessments. They also reveal differences between manually defined and model-generated assessment components, particularly in observation selection and compliance evaluation.

In general, the results demonstrate the feasibility of maintaining a traceable connection between regulatory requirements, standardized security-control context, system evidence, and compliance decisions. They also show that the evidence selected for an assessment can influence the resulting compliance decision, highlighting the importance of evaluating the evidence-grounding process in addition to the final decision.

Sommario

La crescente dipendenza della società moderna dai sistemi informativi ha reso la conformità in materia di cybersicurezza sempre più importante per le organizzazioni, che devono dimostrare che i propri sistemi soddisfano i requisiti di sicurezza previsti. In questo contesto, la valutazione della conformità riveste un ruolo fondamentale. Questa valutazione richiede di collegare requisiti normativi di alto livello a evidenze concrete provenienti dal sistema oggetto di valutazione. Questo processo risulta complesso, poiché le informazioni necessarie per valutare la conformità sono distribuite tra le configurazioni di sistema, le definizioni dell'infrastruttura, lo stato di esecuzione e altri artefatti tecnici. Oltre a produrre una decisione di conformità, queste valutazioni devono mantenere un collegamento chiaro tra le decisioni e le evidenze su cui si basano.

Questo lavoro di tesi presenta una metodologia per la valutazione della conformità nell'ambito della cybersicurezza basata sulle evidenze e che risulta tracciabile. La metodologia proposta organizza la valutazione in tre fasi principali: mappatura dei controlli di sicurezza, contestualizzazione delle evidenze e valutazione della conformità. I requisiti normativi sono mappati su controlli di sicurezza standardizzati, mentre le proprietà rilevanti raccolte dal sistema valutato sono rappresentate come osservazioni strutturate. Il requisito, il relativo contesto dei controlli di sicurezza e le osservazioni selezionate formano un caso di valutazione basato sulle evidenze, che viene valutato mediante regole deterministiche oppure mediante un modello linguistico. La provenienza delle evidenze viene preservata durante l'intero processo per garantire la tracciabilità della decisione risultante.

La metodologia proposta è stata valutata con un'implementazione mirata a stabilirne la fattibilità e con una serie di esperimenti controllati. Quattro configurazioni sperimentali sono state utilizzate per esaminare diverse combinazioni di mappature dei controlli di sicurezza definite da esperti e generate dal modello, selezioni delle osservazioni definite manualmente e generate dal modello, e valutazioni della conformità basate su regole e su modelli linguistici. I risultati mostrano che le variazioni delle proprietà del sistema rilevanti per la sicurezza si riflettono nelle evidenze raccolte e si propagano alle conseguenti valutazioni di conformità. Emergono inoltre differenze tra i componenti della valutazione definiti manualmente e quelli generati dal modello, in particolare per quanto riguarda la selezione delle osservazioni e la valutazione della conformità.

In generale, i risultati dimostrano la fattibilità di mantenere un collegamento tracciabile tra requisiti normativi, contesto dei controlli di sicurezza standardizzati, evidenze di sistema e decisioni di conformità. Mostrano inoltre che le evidenze selezionate per una valutazione possono influenzare la conseguente decisione di conformità, evidenziando l'importanza di valutare il processo di contestualizzazione delle evidenze oltre alla decisione finale.

Contents

1	Introduction	1
2	Foundations of Artificial Intelligence	3
2.1	Introduction	3
2.2	Machine Learning	4
2.3	Large Language Models	5
2.3.1	Limitations of Large Language Models	6
2.3.2	Grounding Language Models with External Context	7
2.4	Explainable Artificial Intelligence	7
2.4.1	Taxonomy of XAI	9
2.4.2	XAI Frameworks	10
2.4.3	Evidence Traceability and Auditability	11
3	Foundations of Cybersecurity Compliance	13
3.1	Cybersecurity	13
3.2	NIST Cybersecurity Framework	14
3.3	Cybersecurity Compliance	16
3.4	Security Controls and Regulatory Frameworks	17
4	State of the Art	20
4.1	Rule-Based and LLM-Based Approaches in Cybersecurity	20
4.1.1	Rule-Based Approaches	20
4.1.2	LLM-Based Approaches	21
4.1.3	Comparative Analysis	22
4.2	Explainability, Faithfulness, and Attribution in LLM-Based Systems	23
4.3	Compliance-as-Code and Automated Control Assessment	25
4.4	AI-Driven Compliance Assessment	27
5	Methodological Approach	31
5.1	Cybersecurity Compliance Assessment	31
5.2	Security-Control Mapping	34
5.2.1	Expert Mapping	34
5.2.2	Language-Model-Based Mapping	35

5.3	Evidence Grounding	35
5.3.1	System Evidence Collection	36
5.3.2	Observation Selection	36
5.4	Compliance Evaluation	37
5.4.1	Rule-Based Evaluation	38
5.4.2	LLM-Based Evaluation	38
5.5	Evaluation Metrics	39
6	Experimental Setup	44
6.1	Framework Implementation	44
6.2	Reference System	48
6.3	Experimental Dataset	51
6.4	Controlled Reference System States	53
6.5	Model Configuration	54
6.6	Experimental Configurations	56
6.7	Output and Validation	57
7	Experimental Results	59
7.1	Effects of the Controlled Reference System States	59
7.2	Rule-Based and Language-Model-Based Evaluation	62
7.3	Security-Control Mapping Evaluation	66
7.4	Observation-Selection Evaluation	67
7.5	Evidence Traceability and Faithfulness	67
7.6	Evaluation of Remediation Coverage and Relevance	69
8	Conclusion	71
A	Language-Model Prompts	73
A.1	Security-Control Mapping Prompt	73
A.2	Observation-Selection Prompt	74
A.3	Compliance-Evaluation Prompt	75
A.3.1	System Prompt	75
A.3.2	User Prompt	75
	Bibliography	77

List of Figures

2.1	Principles of XAI introduced by NIST.	8
5.1	Main components and execution flow of the proposed methodology	33
6.1	Architecture of the reference system in state S0	49
7.1	Reference agreement of the language-model-based evaluator per reference system state	64

List of Acronyms

3GPP	3rd Generation Partnership Project
AC	Access Control
AI	Artificial Intelligence
AIS	Attributable to Identified Sources
ATT&CK	Adversarial Tactics, Techniques, and Common Knowledge
AU	Audit and Accountability
AWS	Amazon Web Services
BERT	Bidirectional Encoder Representations from Transformers
CEM	Contrastive Explanation Method
CI	Continuous Integration
CIA	Confidentiality, Integrity, and Availability
CIS	Center for Internet Security
CM	Configuration Management
CMMC	Cybersecurity Maturity Model Certification
CoT	Chain-of-Thought
CP	Continuity Planning
CPU	Central Processing Unit
CSF	Cybersecurity Framework
CVE	Common Vulnerabilities and Exposures
DA	Data Protection
EU	European Union

GDPR	General Data Protection Regulation
Grad-CAM	Gradient-weighted Class Activation Mapping
GRC	Governance, Risk, and Compliance
GRU	Gated Recurrent Unit
HIPAA	Health Insurance Portability and Accountability Act
HTTP	Hypertext Transfer Protocol
IAM	Identity and Access Management
IDS	Intrusion Detection System
IR	Incident Response
ISO	International Organization for Standardization
JSON	JavaScript Object Notation
KL-LUCB	Kullback–Leibler Lower/Upper Confidence Bound
LIME	Local Interpretable Model-agnostic Explanations
LLM	Large Language Model
LMA	Logging, Continuous Monitoring and Audit
LORE	Local Rule-based Explanations
LSTM	Long Short-Term Memory
ML	Machine Learning
MLOps	Machine Learning Operations
NATO	North Atlantic Treaty Organization
NIST	National Institute of Standards and Technology
NIST IR	NIST Interagency/Internal Report
NLP	Natural Language Processing
OMB	Office of Management and Budget
OPA	Open Policy Agent
OSCAL	Open Security Controls Assessment Language
PCI DSS	Payment Card Industry Data Security Standard
POS	Protection of Services

PSW	Protection of Software
RAG	Retrieval-Augmented Generation
RNN	Recurrent Neural Network
SC	System and Communications Protection
SHA-256	Secure Hash Algorithm with a 256-bit digest
SHAP	SHapley Additive exPlanations
SOC	System and Organization Controls
SP	Special Publication
TLS	Transport Layer Security
XAI	Explainable Artificial Intelligence
XML	Extensible Markup Language
YAML	YAML Ain't Markup Language

List of Tables

4.1	Comparison of Rule-Based and LLM-Based approaches based on [45], [46].	23
5.1	Metrics for evidence traceability and faithfulness	41
5.2	Metrics used to evaluate remediation output	42
6.1	Functionalities of the main software components of the proposed experimental framework	45
6.2	Main technologies and components of the reference system	50
6.3	Runtime configuration of the reference system in state S0	51
6.4	Distribution of D/0048 regulatory requirements used in the experiment	52
6.5	Distribution of system-observation values across reference system states	53
6.6	Controlled reference system states used in the experiments	54
6.7	Configuration of the model-based tasks using Qwen2.5-1.5B-Instruct	55
6.8	Decision logic implemented by the rule-based evaluator	56
6.9	Configurations used in the experiments	56
7.1	Evidence-derived reference decisions for the expert mapping and manually defined observation selection assessment path	60
7.2	Reference-decision changes caused by the controlled system modifications with respect to the baseline	60
7.3	Aggregate compliance evaluation results across the five reference system states	62
7.4	Reference agreements of the language-model-based evaluator across system states	63
7.5	Aggregate confusion matrix for Configuration B	64
7.6	Aggregate confusion matrix for Configuration D	64
7.7	Comparison of model-generated and expert-defined security-control mappings	66
7.8	Aggregate evidence traceability and faithfulness metrics	68
7.9	Aggregate remediation evaluation metrics for the language-model-based configurations	69

Chapter 1

Introduction

The increasing dependence of modern society on information systems has made cybersecurity an essential component for protecting information, services, and critical infrastructures. In fact, information systems are continuously exposed to many diverse security threats. At the same time, the infrastructures supporting these systems are becoming more complex, interconnected and difficult to be properly assessed. Organizations must not only implement appropriate security measures, but also demonstrate that their systems satisfy the cybersecurity requirements imposed by applicable regulations, directives, standards, and organizational policies.

Cybersecurity compliance provides a way to assess whether security requirements are being addressed. In practice, performing such an assessment is not straightforward. Regulatory requirements are generally written for human interpretation and describe expected security properties at a relatively high level. The technical information needed to assess these requirements is provided at a different level, distributed across system configurations, infrastructure definitions, runtime state, and other system artifacts. A regulatory requirement must therefore be interpreted and connected to security-relevant properties that can actually be observed in the system.

These issues create an important challenge for compliance automation of information systems. Automating the interpretation of regulatory text does not by itself establish whether a requirement is satisfied by the system being assessed. An automated assessment must also determine which system evidence is relevant, preserve where that evidence originates, and connect it to the resulting compliance decision. Without this connection, a decision or explanation may be produced without a clear way of determining how it relates to the actual state of the system.

Rule-based approaches provide deterministic and auditable decisions when the assessment logic is explicitly defined. Language models, on the other hand, provide greater flexibility for interpreting natural-language requirements and security context and can additionally generate explanations and remediation guidance. Existing work on compliance automation has explored these and other

approaches through information extraction, semantic similarity, classification, regulatory reasoning, and automated policy assessment. However, most of these works focus on legal documents, organizational policies, or regulatory reasoning over textual information. Comparatively less attention has been dedicated to connecting natural-language cybersecurity requirements with standardized security-control context and evidence collected from the system being assessed, while preserving the traceability of the resulting decision.

Motivated by this research gap, this thesis work presents a methodology for evidence-grounded and traceable cybersecurity compliance assessment. The proposed approach connects regulatory requirements with a standardized security-control context and with system observations obtained from the assessed environment. The regulatory requirement, its security-control context, and the selected system observations are combined into a grounded assessment case. This provides an explicit path from the original requirement to the system evidence used in its assessment and, finally, to the resulting compliance decision.

The proposed methodology is assessed through a proof-of-concept implementation and a set of controlled experiments. The experimental evaluation uses selected regulatory requirements and a containerized reference system configured in five controlled states. Four experimental configurations are considered to examine different combinations of expert and model generated security-control mappings, manual and model generated observation selections, and rule-based and language-model-based compliance evaluation. The experiments are designed to examine how changes in the assessed system are reflected in the collected evidence and resulting compliance decisions, while also evaluating mapping, observation selection, evidence traceability, faithfulness, and remediation.

The remainder of this thesis is organized as follows: Chapter 2 provides the artificial intelligence concepts relevant to this work, with particular attention to language models and explainability. Chapter 3 introduces the cybersecurity compliance concepts and standards that form the technical and regulatory basis of the thesis. Chapter 4 reviews existing approaches to cybersecurity and compliance automation and discusses the research context in which the proposed work is positioned. Chapter 5 presents the methodology proposed in this thesis for connecting regulatory requirements with security-control context and system evidence, and for producing traceable compliance assessments. Chapter 6 describes the implementation of the methodology and the experimental environment used to evaluate it. Chapter 7 presents and analyzes the experimental results, including the effects of the controlled system states, compliance-evaluation results, security-control mapping and observation selection, evidence traceability and faithfulness, and remediation output. Finally, Chapter 8 summarizes the main findings of the thesis work and discusses directions for future research.

Chapter 2

Foundations of Artificial Intelligence

Artificial Intelligence is one of the key technologies that underlie contemporary software systems. Methods in this domain enable software to analyse information, solve complicated issues, and learn from experience. Within this field, machine learning enables systems to learn patterns from data, while recent language models extend these capabilities to the interpretation and generation of natural language. These developments are relevant when textual requirements and technical information must be analysed together. On the other hand, the use of language models introduces limitations related to reliability, traceability, and the relationship between generated outputs and their supporting information. This chapter provides a broad overview of the artificial intelligence concepts used in this thesis work.

2.1 Introduction

Artificial Intelligence (AI) is a field that seeks to understand human intelligence and develop intelligent systems that can behave similarly. Today, AI includes a broad range of applications, such as learning, reasoning, perception, and problem-solving. Additionally, it supports a wide variety of applications, including playing chess, performing mathematical proofs, generating natural language, driving autonomous vehicles, and diagnosing diseases.

According to Russell and Norvig, AI research has historically been based on two fundamental perspectives [1]. The first perspective aims to develop systems that mimic human thinking and behavior patterns. The second, based on the concept of rationality, aims to design systems that can take the most appropriate action in a given situation. Similarly, some approaches explain intelligence through internal thought processes, while others focus only on observable behaviors. These different perspectives have contributed to the diversification of methods and research approaches used in the field of AI over time.

An approach based on human behavior is the Turing Test, proposed by Alan Turing [2]. According to this approach, a computer can be considered intelligent if it can produce answers during written communication that are indistinguishable from those of a human interrogator. Developing such a system requires the combined work of fundamental AI components such as natural language processing, information representation, automated reasoning, and machine learning. In an extended version of the test that includes interaction with the physical environment, fields such as computer vision and robotics are also involved in the process. Nevertheless, Russell and Norvig emphasize that the fundamental aim of AI research is not only to develop systems that can pass the Turing test, but also to understand the fundamental principles underlying intelligence.

Another approach used in the field of AI is based on the concept of rationality. According to this approach, the goal of AI systems is not to perfectly imitate human behaviour, but to perform the most appropriate action in a given situation. Russell and Norvig define such systems as rational agents. A rational agent is a system that perceives its environment, acts autonomously toward specific goals, adapts to changes in its environment over time, and attempts to make decisions that yield the best expected outcome under uncertainty. This approach has become the most widely accepted paradigm in AI research today since it does not focus only on logical inference processes, but also encompasses probabilistic reasoning and learning mechanisms.

2.2 Machine Learning

Machine learning (ML) is a fundamental subfield of AI that aims to develop algorithms capable of improving their performance on specific tasks over time by learning from data. Referring to Mitchell's widely accepted definition [3], Goodfellow et al. [4] state that a computer program begins to learn when it improves its performance on a given set of tasks according to a specific performance metric as it gains experience. This definition shows that the fundamental goal of machine learning is to improve system behavior based on the experience gained from the data, rather than applying predefined rules.

ML approaches offer significant advantages in solving problems that are difficult or impossible for humans to solve using fixed rules. Image recognition, speech recognition, natural language processing, and anomaly detection are just some of the many areas where ML is commonly used. These systems can generalize to previously unfamiliar data by learning patterns from examples, and can offer effective solutions to real-world problems. Therefore, the primary goal of ML is not only to correctly learn training data but also to successfully transfer the learned information to new and unseen data.

In the literature, ML algorithms are generally subdivided into two main categories: supervised and unsupervised learning. In supervised learning, the model is trained using labeled datasets that include both input data and correct output, and the main goal is to learn the relationship between input and output to generate accurate predictions for new data. On the other hand, in unsupervised

learning, the training data does not contain any label. In this case, algorithms try to discover patterns, similarities, or probability distributions within the data. Furthermore, different learning paradigms such as semi-supervised learning and multi-instance learning are also used in some applications, but supervised and unsupervised learning are considered the most fundamental approaches to ML.

Advances in ML allowed the emergence of deep learning methods that facilitate the processing of more complex data structures. While traditional ML methods often require features to be defined by experts, deep learning models can learn these representations directly from data through multilayer neural networks. This ability to learn hierarchical feature representations directly from data significantly expanded the range of problems that could be addressed using machine learning [5].

2.3 Large Language Models

Large Language Models are based on Transformer language models that have been pre-trained on large-scale text collections and possess the ability to understand and generate natural language [6]. LLMs represent the latest stage in the evolution of language modeling. They emerged following statistical language models, neural language models, and pre-trained language models. Although previous generations of language models were mostly designed for use in specific Natural Language Processing (NLP) tasks, LLMs are being developed as general-purpose models capable of performing a much wider range of tasks.

The Transformer architecture, proposed by Vaswani et al. in 2017, forms the basis of many LLMs used today [7]. Before the introduction of the Transformer, NLP models relied primarily on Recurrent Neural Networks (RNNs) and their variants, including Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) architectures, to process sequential data. Since these models processed the input sequence step by step, each computation was dependent on the previous step. Therefore, the training process had to be carried out sequentially, and the possibility of parallel computation was significantly limited, especially with long sequences. This increased training time and made it more difficult to capture long-range dependencies.

The Transformer architecture was developed to overcome these limitations. Recurrent and convolutional layers are completely removed, and the architecture is built entirely around attention mechanisms. As a result, all tokens in an input sequence can be processed simultaneously, which allows the model to capture dependencies between distant tokens in the sequence. This design significantly improves training parallelization while also achieving higher translation quality than previous architectures [7].

The Transformer follows the conventional encoder-decoder architecture consisting of an encoder and a decoder. However, unlike previous approaches, both the encoder and decoder layers consist of multi-head self-attention mechanisms and position-wise feed-forward networks instead of recurrent neural networks. The encoder creates a contextual representation of the input sequence, while

the decoder uses these representations to generate the output sequence step by step. Residual connections and layer normalization are used in each layer to ensure more stable training.

In addition to the architectural advantages provided by the Transformer architecture, scaling has become one of the key factors driving the success of LLMs. The literature indicates that scaling model parameters, training data, and computational resources together not only improve performance on existing tasks but also reveals new capabilities not observed in smaller models [8]. These capabilities, referred to as *emergent abilities*, describe behaviors that naturally emerge once a model reaches a sufficient scale rather than being explicitly programmed. As a result, LLMs can generalize across a wide range of natural language processing tasks without requiring task-specific retraining. One of the most notable examples is *in-context learning*, which enables models to perform new tasks using only instructions or a few demonstrations provided within the prompt [6]. Furthermore, models can follow instructions given in natural language and perform multi-step reasoning by creating intermediate reasoning steps in complex problems. In particular, the Chain-of-Thought (CoT) prompting approach has been shown to substantially improve performance on tasks involving mathematical inference, logical reasoning, and complex decision-making. Collectively, these capabilities have transformed LLMs from text generation models into general-purpose systems capable of contextual understanding, knowledge transfer across tasks, and sophisticated reasoning.

2.3.1 Limitations of Large Language Models

Despite their capabilities across different natural-language tasks, LLMs also have limitations that need to be considered when their outputs are used for assessments. One of these limitations is hallucination. LLMs may generate content that appears plausible but deviates from the provided input, conflicts with previously generated context, or is inconsistent with established factual knowledge [9]. The apparent plausibility of such outputs can also make these errors difficult to identify.

The output of an LLM can also be affected by how the task is presented to the model. Variations in prompt design can lead to different results, while the choice of decoding parameters can further affect model performance [10]. These factors create reproducibility concerns when the details of prompt construction, model versions, and response generation settings are not clearly documented.

These limitations are particularly relevant to cybersecurity and compliance-related tasks. LLM performance can be sensitive to the prompting strategy, while the lack of interpretability and traceability can make it difficult to determine the rationale behind an output [11]. In addition, errors in compliance assessment are not limited to hallucination. A model may retrieve the correct information but fail to reason over it correctly, or it may fail to identify relevant information that is already available in the provided context.

These issues are very relevant when an assessment depends on specific technical or regulatory information. Supplying relevant information to the model

can reduce some limitations, but the presence of that information does not guarantee that it will be correctly identified or used in the resulting assessment.

2.3.2 Grounding Language Models with External Context

Language models can be provided with external context as part of their input rather than relying only on knowledge represented in their model parameters. Such context can contain information that is dynamic, specialized, or otherwise not available to the model through its existing knowledge. Grounding concerns the relationship between the provided context and the response generated by the model.

Producing a correct answer does not necessarily mean that the response is fully grounded in the provided information. Lee et al. [12] distinguish between answer correctness and a stricter notion of grounding. Under this definition, a grounded response uses the necessary knowledge contained in the provided context while remaining within the scope of that context. Therefore, a response may contain the expected answer while still omitting relevant contextual information or introducing information that does not originate from the provided context.

External context is also relevant when language models are applied to cybersecurity compliance tasks. Security assessment may require information in different forms of representation. For example, structured descriptions of system behaviour can be considered together with external security knowledge expressed in standards or regulatory documents. Structured process models are combined with external security standards to support context-aware explanation and compliance checking, illustrating how external information can provide the security context needed to assess structured system information.

Providing relevant context does not ensure that all this information will be correctly used. In compliance-oriented experiments, errors occur when a language model fails to identify relevant information already present in the context or retrieves the appropriate information but fails to connect it correctly during reasoning. Therefore, grounding requires attention not only to the information supplied to the model, but also to whether the resulting output remains connected to that information.

2.4 Explainable Artificial Intelligence

Although artificial intelligence systems are increasingly used across a wide range of applications, a significant portion of these systems have complex and opaque decision-making mechanisms [13]. While deep learning-based models offer high accuracy, their internal operating principles function as black-box systems, making them difficult for humans to understand [14]. This problem is even more important in the context of cybersecurity, as decisions made in this area directly affect system security, data integrity, and organizational risks. A system's inability to explain its decisions leads to a lack of trust and makes it difficult for

security professionals to adopt these systems. Therefore, there is a need for artificial intelligence systems that not only produce accurate results but also explain these results in a meaningful way. This need is addressed under the concept of eXplainable AI (XAI).

In this regard, a critical framework for explainable AI systems has been established by the National Institute of Standards and Technology (NIST) [15]. NIST defines four fundamental principles for explainable AI systems: *Explanation*, *Meaningfulness*, *Explanation Accuracy*, and *Knowledge Limits* as shown in Figure 2.1.

The explanation principle requires systems to provide a rationale or supporting information for each output, while the meaningfulness principle means that these explanations must be understandable to the user. Explanation accuracy mandates that the explanations provided accurately reflect the model’s actual decision-making process, while the knowledge limits principle requires systems to be able to identify situations where they are unreliable and to know their own limits of authority.

These principles demonstrate that explainability is not only a technical feature, but also a critical requirement in terms of trust, auditability, and regulatory compliance. Particularly, in the context of cybersecurity, these principles enhance the traceability of decision-making processes and ensure that systems are used more reliably.



Figure 2.1: Principles of XAI introduced by NIST.

2.4.1 Taxonomy of XAI

Over the years, numerous methods and approaches have been developed to improve the explainability of artificial intelligence models, resulting in a comprehensive XAI taxonomy.

According to this taxonomy, one of the fundamental distinctions is between transparent models and post-hoc explanatory methods [16]. Transparent or interpretable models refer to systems whose decision-making mechanisms can be directly understood by humans. In contrast, post-hoc approaches are additional techniques developed to explain the behavior of complex black-box models after training. These methods aim to provide interpretability without changing the model structure.

Another important classification is based on model dependence. In this context, XAI methods are divided into model-specific and model-agnostic methods. Model-specific methods are developed for specific machine learning models and generally require access to internal information such as the model’s weights or structural parameters. In contrast, model-agnostic methods can be applied to different machine learning models without requiring access to the model’s internal structure. This flexibility enables model-agnostic methods to be widely used in various application areas.

XAI methods are also classified according to the scope of the explanation. Global explanation methods aim to explain the overall behavior of the model and the decision-making logic, while local explanation methods focus on the estimated explanation generated for a specific sample. Global explanations are generally used to understand the behavior of the model across the entire dataset, while local explanations reveal why a particular decision was taken.

Furthermore, intrinsic and extrinsic explainability approaches are also defined. Intrinsic approaches refer to models that are inherently interpretable, such as decision trees or rule-based systems. Extrinsic or post-hoc approaches, on the other hand, rely on the application of additional explanation mechanisms after training to enable the interpretation of complex models.

Explainability is not only a technical characteristic: it is related to concepts such as transparency, fairness, trust, usability, reliability, and causality. Transparency refers to a system’s decision-making process being understandable and traceable by the user. Fairness aims for the system not to discriminate between different users or data groups and to produce unbiased decisions. Trust reflects the confidence and comfort of users when interacting with and relying on the system, while usability focuses on explanations being easily understood and used by the target user. Reliability describes how resistant a system is to variations in its inputs and parameters. Finally, causality aims for the model to be able to explain not only relationships but also cause-and-effect connections. Especially in cybersecurity applications, it is of great importance for security experts not only to obtain accurate predictions but also to understand why these decisions were made. Therefore, explainability is considered a critical requirement for building trust and supporting human-machine collaboration.

2.4.2 XAI Frameworks

XAI frameworks are tools and methods aimed at explaining how artificial intelligence models work and generating interpretable information about model behavior. The frameworks address different explanatory needs and differ in terms of interpretability scope, model dependency, and explanation methodology.

Local Interpretable Model-agnostic Explanations (LIME) is a framework that aims to generate local explanations for individual predictions [17]. It is considered a model-agnostic method because it does not require access to the internal structure of the model. This method observes the model’s behavior by modifying the input data around a specific data sample. Then, an interpretable model is trained on the newly generated samples to determine which features contribute most to the prediction. LIME is widely used, particularly for interpreting complex black-box models at the local level.

Shapley Additive exPlanations (SHAP) is another model-independent explanatory method that focuses on calculating the contribution of each feature to a given prediction [18]. This approach is based on Shapley values in coalition game theory. In the SHAP method, each feature is treated like a player contributing to the prediction, and the contribution of each feature to the model is calculated mathematically. Thus, it is possible to explain more systematically and fairly which features are more effective on a given prediction. Similar to LIME, SHAP is considered a model-agnostic and post-hoc approach. However, unlike the LIME method, tree-specific SHAP approaches such as TreeSHAP do not require the creation of a local explanation model; instead, they use the same mathematical function based on the tree structure to calculate the Shapley values for each feature dimension.

The Anchors approach generates decision rules in an IF-THEN structure to explain a specific prediction [19]. The goal of this method is to determine under what conditions the model maintains a particular decision without changing it. In other words, if changes in other features do not alter the model output while certain features are preserved, those features are considered "anchors" for the prediction. To efficiently balance exploration and exploitation during rule generation, the Anchors method frames the problem as a multi-armed bandit formulation using the KL-LUCB algorithm, combined with a beam search algorithm to identify the optimal rule with a minimum number of model queries.

Local Rule-based Explanations (LORE) is a post-hoc and model-agnostic explanation framework that aims to generate rule-based local explanations for black-box models [20]. In this method, artificial data samples are first created, and then an interpretable decision tree is trained using these samples. The trained decision tree is used to extract rules explaining the current decision and counterfactual explanations showing how a different decision could be obtained.

Other domain-specific explanation methods have been proposed for particular machine learning tasks. For example, Gradient-weighted Class Activation Mapping (Grad-CAM) provides visual explanations for convolutional neural networks by highlighting the image regions that contribute most to a prediction [21], whereas CEM focuses on generating comparative explanations by identifying the

features that support or contradict a classification outcome [22]. Although these methods have demonstrated effectiveness within their respective application domains, they are generally designed for specific model types and therefore have more limited applicability than model-agnostic explanation frameworks such as LIME and SHAP.

2.4.3 Evidence Traceability and Auditability

Explainability is commonly concerned with making the behaviour and outputs of AI systems understandable. Post-hoc methods such as LIME and SHAP address this problem by providing information about individual predictions and the influence of their inputs. However, explaining an individual output does not necessarily provide the information required to reconstruct how a decision was reached.

In applications where decisions must be reviewed after they have been produced, it is also important to preserve the connection between a decision and the evidence that supports it [23]. This leads to the concept of evidence traceability. Evidence traceability concerns whether the factors involved in a decision can be connected to their supporting evidence through a documented path. The focus is therefore broader than the explanation of an isolated model output. It also includes the preservation of information that allows the basis of a decision to be inspected afterwards. A traceable decision record can support this process by retaining links between the information available to the system and the decision produced from it.

Provenance and lineage provide related mechanisms for maintaining these links [24]. Provenance records information about the origin and processing history of technical artifacts, while lineage represents relationships between artifacts across the processes that produce and transform them. In AI systems, these mechanisms can be used to connect outputs with input data, transformations, model information, and operational records. Preserving such relationships creates a reproducible technical trail that can be inspected to determine how the information associated with a particular output was obtained and processed.

The preservation of this technical path is closely related to auditability. An explanation may provide useful information about a model output without, by itself, creating a complete record that can be verified afterwards. For this reason, explainability artifacts can be complemented by provenance information, audit logs, and other structured evidence. Connecting these elements allows explanations to be associated with technical records that support later verification rather than remaining isolated descriptions of model behaviour [25]. This relationship becomes particularly important in regulated settings, where technical evidence may need to be linked to requirements and retained in a form that supports inspection and oversight.

Evidence and context are also important when language models are used for security-related assessment tasks. Language-model outputs can be sensitive to the prompting strategy, and their limited interpretability and traceability can make the basis of an output difficult to determine. They may also generate

plausible information that is not correct. One way of addressing these limitations is to provide external information that gives the model additional context for the task. In cybersecurity assessment, this context can include structured descriptions of the system together with external security knowledge used to interpret them [11].

Providing relevant context does not by itself ensure that an assessment is correct. Errors can also occur when relevant information is not retrieved or when available information is not correctly combined during the assessment process. It is therefore useful to distinguish between the availability of supporting information and the ability to preserve and inspect its relationship with the resulting output.

Consequently, evidence traceability complements conventional explainability. Post-hoc explanation methods primarily provide information about model behaviour or individual predictions, whereas evidence traceability concerns the documented relationship between an output and the information supporting it.

Chapter 3

Foundations of Cybersecurity Compliance

Cybersecurity can be defined as the combination of technologies, processes, practices, and protective measures used to safeguard information systems, networks, and data against cyberattacks, unauthorized access, damage, and disruption, while preserving the confidentiality, integrity, and availability of information [26], [27]. Cybersecurity also involves managing security risks, establishing governance processes, meeting regulatory requirements, and regularly reviewing and improving security controls as threats and organizational requirements change. Organizations are not only required to implement proper cybersecurity measures, but they are also supposed to prove that the cybersecurity measures being put in place meet relevant regulations and specific organizational requirements. In this chapter, the concept of cybersecurity compliance is analyzed by exploring its fundamental concepts.

3.1 Cybersecurity

As modern society, critical infrastructure, and economies have become increasingly reliant on computer networks and information technology, the importance of cybersecurity has increased [28]. This increasing dependence has also made cyber attacks more attractive to adversaries and extended their potential consequences. As a result, cyber attacks have continued to increase in both frequency and sophistication, highlighting the need for effective cybersecurity strategies to protect digital systems and information assets.

Cybersecurity is generally concerned with understanding cyber threats and developing appropriate defensive measures that protect digital systems and information assets against unauthorized activities. A fundamental objective of cybersecurity is maintaining the three well-established security principles commonly known as the CIA triad: Confidentiality, Integrity, and Availability [29]. Confidentiality refers to preventing unauthorized disclosure of information, in-

egrity ensures that information cannot be modified or deleted without authorization, while availability guarantees that systems and information remain accessible to authorized users whenever needed. These three security objectives constitute the fundamental principles upon which cybersecurity mechanisms and protection strategies are established.

Malicious software, commonly known as malware, has become one of the primary instruments used by attackers to compromise digital systems. Malware represents a broad category of attacks including spyware, viruses, Trojan horses, worms, and bot executables. They are designed to compromise systems without the owner's knowledge and perform unauthorized activities benefiting an attacker. Malware may spread through infected devices, malicious websites, deceptive email attachments, removable media, or compromised network services, enabling adversaries to target end-user devices, enterprise servers, networking equipment, and even industrial control systems.

Historically, organizations mainly relied on perimeter defense models to defend their information systems. This traditional approach focused on protecting the organizational boundary through technologies such as firewalls, intrusion detection systems, antivirus software, and access control mechanisms. Incoming network traffic was inspected before reaching internal resources, while authentication and authorization mechanisms limited access to authorized users. Accountability mechanisms were also incorporated to detect misuse and support incident investigations. The effectiveness of traditional perimeter strategy protection has gradually decreased as malware has evolved. Attackers have developed sophisticated techniques capable of bypassing conventional security controls by exploiting vulnerabilities across hardware, software, and network layers or by abusing emerging technologies such as cloud computing, social media platforms, smartphones, and critical infrastructure. Consequently, cybersecurity defenses must continuously adapt to address emerging threats and increasingly sophisticated attack techniques.

3.2 NIST Cybersecurity Framework

As cybersecurity risks have become increasingly complex, a structured approach to manage and communicate those risks is required. The NIST CSF 2.0 provides guidance to manage cybersecurity risks by defining a taxonomy of high-level cybersecurity outcomes [30]. The framework is intended for organizations of all sizes, sectors, and levels of cybersecurity maturity, and may be used to understand, assess, prioritize, and communicate cybersecurity efforts. Instead of describing how cybersecurity outcomes should be achieved, the CSF refers users to complementary resources on practices and controls that can be used to achieve the outcomes.

The framework consists of three main components: CSF Core, Organizational Profiles, and Tiers. The CSF Core defines a taxonomy of cybersecurity outcomes that can be applied to manage cybersecurity risks. Organizational Profiles describe an organization's current or target cybersecurity posture in

terms of these outcomes, while Tiers describe the level of rigor applied to cybersecurity risk governance and management practices within an organization.

The CSF Core organizes cybersecurity outcomes into a hierarchy of Functions, Categories, and Subcategories. These outcomes are not intended to serve as a checklist of actions because the specific measures required to achieve them depend on the organization and its operational context. Likewise, the ordering of Functions, Categories, and Subcategories does not represent either their implementation sequence or their relative importance. Instead, the structure is designed to support individuals responsible for operationalizing cybersecurity risk management within an organization.

At the highest level, the CSF Core is organized into six Functions:

- *Govern*: Establishes the organization’s cybersecurity risk management strategy, expectations, and policies, and provides the context for the remaining Functions.
- *Identify*: Focuses on understanding organizational assets, suppliers, and cybersecurity risks.
- *Protect*: Defines safeguards that reduce the likelihood and impact of cybersecurity events.
- *Detect*: Addresses the timely discovery and analysis of potential cybersecurity attacks and incidents.
- *Respond*: Covers actions taken after cybersecurity incidents have been detected.
- *Recover*: Focuses on restoring affected assets and operations following a cybersecurity incident.

Although each Function addresses a different aspect of cybersecurity risk management, the framework states that they should be implemented concurrently rather than sequentially.

The framework introduces Organizational Profiles and Tiers to support the implementation of the CSF Core. Each organization creates at least one or both of the following:

- *Current Profile*: Describes the cybersecurity outcomes that an organization is currently achieving.
- *Target Profile*: Represents the outcomes selected to achieve its cybersecurity risk management objectives.

Comparing the Current and Target Profiles helps organizations to identify gaps between their current and desired cybersecurity outcomes and to develop prioritized action plans for continuous improvement. CSF Tiers provide further information about the maturity level of cybersecurity risk governance and management practices of an organization, ranging from *Partial* (Tier 1) to *Adaptive* (Tier 4). The Tiers complement an organization’s existing risk management methodology rather than replacing it.

3.3 Cybersecurity Compliance

Cybersecurity compliance refers to the process by which organizations ensure that their security practices, policies, and technical controls comply with applicable laws, regulations, industry standards, and internal requirements [31]. In modern digital environments, organizations are required to protect sensitive data, maintain secure information systems, and demonstrate that their security measures align with relevant legal and regulatory obligations. Therefore, compliance is not limited to satisfying formal requirements but it also supports the establishment of baseline security practices to reduce the likelihood of breaches and to protect critical assets [32].

Within the broader cybersecurity context, compliance is closely related to the concepts of Governance, Risk, and Compliance (GRC) [33].

- *Governance*: Ensures that organizational activities are aligned with strategic objectives.
- *Risk Management*: Focuses on identifying, assessing, and mitigating potential threats.
- *Compliance*: Ensures adherence to applicable laws, regulations, and standards.

From this perspective, cybersecurity compliance plays a central role in connecting regulatory obligations with operational security practices. Effective compliance requires organizations not only to meet legal requirements, but also to integrate these requirements into their cybersecurity processes and maintain appropriate security measures over time [34].

It is also important to distinguish cybersecurity compliance from cybersecurity itself. Cybersecurity refers to the broad set of activities that aim to protect information systems against threats, including risk management, incident response, threat intelligence, and continuous monitoring [35]. Compliance, on the other hand, focuses on demonstrating that specific requirements have been satisfied. Although these two concepts are different, compliance can provide a foundation for cybersecurity by requiring organizations to implement the necessary controls, document security practices, and meet legal obligations. For this reason, compliance should not be treated as a one-time checklist activity, but rather as an ongoing commitment to pursuing security.

Security compliance provides a structured basis for assessing whether an organization's security practices satisfy specified requirements [31]. In practice, this involves translating regulatory, legal, or standards-based requirements into organizational controls and examining whether these controls are implemented and maintained. Activities such as risk assessments, policy reviews, audits, incident response planning, and continuous monitoring can provide evidence for this assessment. Compliance should therefore not be considered equivalent to cybersecurity itself; rather, it provides a mechanism for evaluating and documenting whether defined security requirements are being addressed within the organization.

Cybersecurity compliance also presents several challenges. Organizations often need to comply with multiple frameworks and standards at the same time, such as NIST CSF, System and Organization Controls 2 (SOC 2), Center for Internet Security (CIS) Controls, General Data Protection Regulation (GDPR), Health Insurance Portability and Accountability Act (HIPAA), Payment Card Industry Data Security Standard (PCI DSS), and Cybersecurity Maturity Model Certification (CMMC) [36]. Since these frameworks evolve independently, differ in their objectives, scope, and implementation requirements, organizations might face difficulties in interpreting abstract requirements, coordinating compliance activities across different units, and dealing with resource constraints to maintain compliance [37]. Another important challenge is the possible gap between formal compliance and actual security needs. Compliance is sometimes treated as a "check-the-box" activity, where organizations focus primarily on satisfying regulatory requirements rather than addressing their specific security risks. This can create a false sense of security, since satisfying compliance requirements does not always guarantee that an organization is fully protected against evolving cyber threats [38].

3.4 Security Controls and Regulatory Frameworks

The assessment of security compliance begins with identifying the security requirements that are applicable to an information system or an organization [39]. Rather than being defined independently by individual organizations, these requirements are generally established through regulations, directives, policies, standards, and organizational objectives, and are implemented in practice by security controls. According to the Office of Management and Budget (OMB) Circular A-130 [40], a *security control* is defined as:

“The safeguards or countermeasures prescribed for an information system or an organization to protect the confidentiality, integrity, and availability of the system and its information.”

Similarly, NIST SP 800-53 describes security and privacy controls as safeguards implemented to protect organizational operations, assets, individuals, and information systems from a diverse range of threats and risks. Furthermore, the NIST standard states that these controls are obtained from mission and business needs, as well as laws, directives, executive orders, regulations, policies, and guidelines.

Rather than isolated requirements, security controls are documented within standards and regulatory directives that specify the security measures to be implemented to protect information and information systems. These documents define mandatory requirements and procedures that organizations are expected to follow. For example, the NATO Directive on the Security of Information establishes mandatory provisions governing the classification, handling, dissemination, transmission, storage, and destruction of NATO information, as well

as procedures for reporting security breaches and releasing classified information [41]. Consequently, compliance assessment involves evaluating whether the security measures implemented in a target system satisfy the requirements specified by the applicable directive or standard.

The growing number of regulatory directives, standards, and organizational policies has increased the complexity of managing security requirements originating from different sources. As a result, representing these security requirements in a structured and consistent manner has become increasingly important to support systematic interpretation and compliance assessment. Standardized control catalogs provide one means of organizing such requirements, while machine-readable representations provide a means of structuring the associated security information for automated processing and exchange. NIST SP 800-53 and Open Security Controls Assessment Language (OSCAL) address these two aspects, respectively.

NIST Special Publication 800-53 is one of the most widely adopted security control catalogs that provides a broad set of security and privacy controls to manage risks for information systems and organizations [39]. Rather than serving only as a checklist of security requirements, the standard presents security controls as part of an extensive approach to protect organizational operations, assets, individuals, and information systems. To facilitate their implementation and assessment, controls are organized into predefined control families, including, among others:

- *Access Control (AC)*,
- *Audit and Accountability (AU)*,
- *Configuration Management (CM)*,
- *Incident Response (IR)*, and
- *System and Communications Protection (SC)*.

This organization enables related security requirements to be managed systematically and supports consistent processes for selecting, implementing, and assessing security controls.

However, defining security controls alone is not sufficient to automate compliance assessment processes. Throughout activities such as security planning, implementation, assessment, authorization, and continuous monitoring, numerous security-related documents are produced which maintain complex relationships with one another [42]. Security plans, assessment plans, assessment results, and policy documents must be traceable across the entire security lifecycle, while assessment findings should be linked to system configurations and the security controls being evaluated.

The work that motivated the development of OSCAL highlights that this documentation ecosystem is inherently complex and security information is primarily maintained in documents intended for human interpretation. Consequently, automatic processing of security information and the ability of interoperability among different security tools remain significant challenges.

To address these challenges, NIST developed the OSCAL format. OSCAL is an open language and data model designed to represent security controls and security assessment information in a machine-readable format. Instead of maintaining security information solely as human-readable documentation, OSCAL provides structured data models that facilitate the validation, exchange, and automated processing of security information across different tools and platforms. To support interoperability, OSCAL adopts a metaschema-based approach capable of generating equivalent representations in XML, JSON, and YAML.

Chapter 4

State of the Art

Recent advances in artificial intelligence, particularly large language models, have increased the use of AI-based techniques in cybersecurity. Research has explored their application to security operations, threat analysis, automated assessment, explainability, and compliance-related tasks. This chapter presents an overview of the state of the art in these areas through a review of recent approaches proposed in the literature.

4.1 Rule-Based and LLM-Based Approaches in Cybersecurity

Artificial intelligence systems in the field of cybersecurity have undergone a major transformation over time [43]. Early AI applications mostly consisted of rule-based systems relying on symbolic logic and predefined rules. While these systems could produce effective results in specific and well-defined tasks, they had critical limitations in adapting to changing threat environments, making contextual inferences, and learning from data. With the development of machine learning and deep learning approaches, AI systems have evolved into structures that can learn from large datasets, improve their performance over time, and analyze more complex patterns. Particularly with the emergence of generative AI systems such as Large Language Models (LLMs), capabilities such as natural language understanding, multi-step reasoning, and tool usage have significantly improved.

4.1.1 Rule-Based Approaches

Rule-based systems are among the earliest applications in the history of artificial intelligence and are often referred to as expert systems. These systems evaluate data and events and produce results through predefined IF-THEN rules. The rule engines in the system match information about the current situation with these rules and trigger the relevant actions when the conditions are met.

Rule-based approaches were applied early on, particularly in intrusion detection systems (IDS), access control mechanisms, and basic network security solutions.

Rule-based applications have several important advantages. First of all, they are strong in terms of interpretability; it is possible to track and understand which rules and logical processes were used to generate decisions. They also have a modular structure; rules can be added or modified independently. Furthermore, specific constraints and requirements can be fully defined and enforced through rules, ensuring high accuracy.

A rule-based cognitive multi-agent architecture was developed in [44] for proactive defense in cybersecurity. In this architecture, each agent focuses on a specific task area and has specialized roles. For instance, a detection agent detects potential threats by using a heuristic approach and symbolic logic, while an analysis agent evaluates and prioritizes these threats. A response agent implements countermeasures against the identified threats, while a learning agent ensures continuous learning of the system by updating the rules with data obtained from past events and inter-agent interactions. These agents collaborate using symbolic inference mechanisms and a shared knowledge base, enabling multi-step analysis and coordinated action processes. Furthermore, the modular structure of the system narrows the decision-making space and increases operational manageability by limiting the scope of work for each agent. In conclusion, cognitive multi-agent architecture offers advantages in terms of explainability and auditability. Since the system transparently presents the rules and logical steps used to generate decisions, it allows security experts and auditors to monitor the decision-making process.

Rule-based systems also present certain limitations. Scalability can be challenging since the number of required rules increases exponentially as domain complexity increases. In terms of adaptability, rules must be manually updated to meet changing requirements. Furthermore, rule-based systems are limited in their ability to handle uncertainty. They may be insufficient in situations requiring probabilistic reasoning. This situation has necessitated the development of adaptive and learning systems against modern cybersecurity threats. Consequently, while rule-based approaches offer the advantage of high explainability and deterministic behavior, they remain limited in terms of flexibility and adaptability, leading to the emergence of data-driven learning-based solutions.

4.1.2 LLM-Based Approaches

In recent years, LLM-based systems have become increasingly relevant to cybersecurity applications. Unlike rule-based systems, these structures can perform more flexible reasoning by combining natural language processing, contextual reasoning, and multi-step planning capabilities, rather than simply applying predefined rules.

One of the fundamental characteristics of LLM-based approaches is multi-step reasoning [45]. Methods such as Chain-of-Thought (CoT) support complex problem solving by structuring the reasoning process into intermediate steps. This reasoning capability can complement the contextual understanding

of LLMs when processing security information and performing tasks that require multiple stages of analysis. The integration of LLMs into agentic frameworks further extends these capabilities through mechanisms such as planning, memory, and tool use, allowing agents to participate in more complex and automated security workflows. Another significant advantage of LLM-based approaches is their ability to generalize to tasks without requiring task-specific retraining for every new problem. Rather than introducing additional model training, the approach is designed to elicit knowledge already embedded within the LLM through role-based collaboration and multi-round reasoning.

Despite these advantages, there are various limitations and security risks of LLM-based systems. First-generation LLM systems (Gen-1) operate in a stateless manner [46]. Since these systems lack persistent memory, they cannot transfer information from one query to another and can only perform reasoning based on the information within the current prompt. Early LLM-based systems mainly rely on internal pattern-matching processes without external verification mechanisms. Therefore, tasks such as log inspection, CVE lookup, or sandbox output analysis cannot be directly validated, which may result in highly confident but incorrect outputs. For this reason, Gen-1 systems are considered useful for supporting tasks such as threat intelligence summarization, explanation generation, and text processing, rather than serving as fully autonomous decision-making systems in high-risk cybersecurity environments. This structure creates significant reliability problems because incomplete or ambiguous prompts may lead the models to generate hallucinations or miss critical details [47]. In addition, even small prompt variations can produce substantially different outputs, which negatively affects reproducibility and limits reliable automation.

In a study by Banerjee et al. [48], it is discussed that the hallucination problem is not merely a temporary error, but a structural feature of the LLM architecture. According to the study, hallucinations cannot be completely eliminated because training data is never entirely complete, information retrieval processes always involve a certain probability of error, and intent classification processes cannot be performed with complete accuracy. The article also states that methods such as Retrieval-Augmented Generation (RAG), Chain-of-Thought (CoT), uncertainty quantification, and fact-checking can reduce the risk of hallucinations but cannot completely eliminate them. It is particularly noted that even fact-checking mechanisms cannot provide 100% accuracy, and models cannot fully predict their own production processes.

4.1.3 Comparative Analysis

Rule-based and LLM-based approaches address different operational requirements in cybersecurity environments. Rule-based systems are particularly effective in scenarios where security policies, workflows, and operational constraints are clearly defined. Since these systems operate through deterministic rules, they provide predictable and auditable behavior, which is especially important for regulatory compliance and enforcing strict security policies. Therefore, rule-based mechanisms continue to play an important role in tasks such as access

control, policy enforcement, and safety constraint implementation.

In contrast, LLM-based systems can support cybersecurity tasks where contextual interpretation and adaptive processing of information are required. Unlike symbolic systems that depend on predefined knowledge, LLM-based architectures can process heterogeneous information sources and support more flexible threat analysis processes. This capability becomes particularly important in modern cybersecurity operations where human analysts must correlate large volumes of alerts, logs, and threat intelligence data under time pressure.

Table 4.1: Comparison of Rule-Based and LLM-Based approaches based on [45], [46].

Feature	Rule-Based Systems	LLM-Based Systems
Decision-Making	Decisions are generated through predefined rules	Decisions can be generated through contextual interpretation of the provided information
Adaptability	Limited adaptability to evolving conditions	Can support adaptation to different task contexts
Applications	Rule-based detection, policy enforcement, and traditional automation	Natural-language analysis, information extraction, decision support, and security automation
Limitations	Limited adaptability to dynamic and evolving conditions	May produce hallucinations and exhibit limitations in interpretability and reliability

The main differences between rule-based and LLM-based approaches are summarized in Table 4.1. Yet, neither approach fully addresses all operational requirements on its own. While rule-based systems may struggle with scalability and adaptation to evolving threats, LLM-based systems introduce challenges related to explainability, reproducibility, and reliability.

4.2 Explainability, Faithfulness, and Attribution in LLM-Based Systems

Recent studies on explainability in Natural Language Processing have increasingly examined whether model explanations faithfully represent the process underlying a prediction. For example, Lyu et al. [49] review more than 110 explanation methods from this perspective and identify faithfulness as a central property of model explanations. In their analysis, an explanation is considered faithful when it accurately reflects the reasoning process behind the corresponding prediction. This property is distinguished from plausibility, since an explanation may appear convincing to a user without accurately representing the mechanism that produced the prediction. The authors emphasize that this distinction is particularly important because plausible but unfaithful explanations can lead users to place unjustified trust in model outputs.

The survey further shows that model explanations can take different forms and operate at different levels. Predictions may be explained through input features, examples, neurons, concepts, or interactions between features, while the resulting explanations may be expressed through importance scores, visualizations, natural language, or causal graphs. Existing approaches are organized into five main families: similarity-based methods, analysis of model-internal structures, backpropagation-based methods, counterfactual intervention, and self-explanatory models. Despite the large number of available methods, the evaluation of faithfulness is identified as an open problem, since there is no generally accepted standard for measuring whether an explanation accurately represents the reasoning process of a model.

Traditional Explainable Artificial Intelligence methods have fundamentally been developed for predictive models that produce a single output from a fixed input. In these settings, explanation techniques such as SHAP and LIME aim to identify the input features that contribute most to an individual prediction. These assumptions become more difficult to apply to agentic AI systems, whose behavior emerges through a sequence of observations, tool invocations, reasoning steps, and state updates rather than a single prediction [50]. Ultimately, the purpose of explanation shifts from feature attribution to the complete execution trajectory. The authors distinguish explainability for static prediction from explainability for agentic systems and emphasize that explanations should account for action sequences, execution context, retrieved evidence, and state evolution. Their experiments further show that although attribution-based methods produce stable explanations in static prediction tasks, they cannot reliably diagnose execution-level failures in agentic workflows. Instead, trace-based explanations based on execution logs are able to localize behavioral failures, such as incorrect tool usage, state inconsistency, and unsuccessful recovery.

In [51], explainability is studied from a complementary perspective by integrating agentic AI with traditional XAI techniques. The proposed Agentic XAI framework combines feature explanations based on SHAP with an LLM that iteratively refines explanations through multiple reasoning rounds. Rather than modifying the underlying prediction model, the framework focuses on improving the quality of the generated explanations by repeatedly incorporating additional analysis and contextual information. The results indicate that iterative refinement significantly improves explanation quality during the early refinement rounds, however, excessive refinement leads to increasingly verbose and less reliable explanations. Based on these findings, the authors identify strategic early stopping as an important consideration for agentic XAI systems, demonstrating that iterative refinement does not necessarily lead to continuous improvements in explanation quality.

Another research direction investigates whether generated statements can be traced to the information that supports them. Rashkin et al. [52] introduce the Attributable to Identified Sources (AIS) framework for evaluating natural-language generation with respect to identified source material. Rather than evaluating generated text only as an isolated output, AIS assesses whether statements concerning the external world can be supported by an independent source

provided to the system. The framework therefore treats attribution as the accurate use of source information to support generated text and relates this problem to hallucination and faithfulness.

The AIS framework was evaluated across conversational question answering, summarization, and table-to-text generation. These settings include both unstructured source documents and structured tabular information, showing that attribution can be examined across different forms of source material. The evaluation is based on determining whether a generated statement is corroborated by the identified source, while complementary properties such as source quality and fluency are treated separately. Human evaluation experiments reported by the authors achieved moderate to high inter-annotator agreement, supporting the feasibility of applying the framework across the considered generation tasks.

4.3 Compliance-as-Code and Automated Control Assessment

Automation of security-control assessment has been addressed by NIST through a structured approach for evaluating implemented security controls. NIST IR 8011 introduces an approach for automating the assessment of controls from NIST SP 800-53 by using the assessment procedures defined in SP 800-53A [53]. In this approach, security controls are decomposed into control items and corresponding determination statements, which define the assessment objectives for the parts of a control being examined. These determination statements can then be verified through defect checks designed to support automated assessment. A defect check is expressed as a test and provides a means of determining whether the corresponding control item satisfies its assessment objective. A central part of this approach is the comparison between the desired state and the actual state of the assessed system. For automated assessment, both states are represented as data so that the assessment system can identify the desired state applicable to an assessment object, obtain the corresponding actual-state value, and compare them without significant human intervention. NIST IR 8011 also recognizes that the extent of automation depends on the control being assessed. While many control assessments can be fully automated, others may only be partially automated or may still require manual or procedural assessment methods.

More recent works have considered compliance-as-code as a means of connecting regulatory obligations, technical controls, and machine-readable evidence. Nweke and Yeng [54] present a compliance-as-code approach for AI-driven identity systems in which clause-level obligations are linked to stable control identifiers, enforcement points, and required evidence types. The approach addresses changes in policies and system configurations by maintaining these relationships as explicit traceability artifacts. The authors also define evidence completeness over fixed monitoring windows using explicit evidence contracts. Missing required evidence is therefore represented as a structured finding rather than remaining an ambiguous absence of information. The implemen-

tation combines policy-as-code enforcement using Continuous Integration (CI) configuration testing, release-time provenance controls, runtime decision logging, and an evidence graph that associates events and resources with control identifiers, releases, and hashes. Assessment information is exported through OSCAL packages, including Component Definition, System Security Plan, Assessment Results, and Plan of Action and Milestones artifacts. Evidence resources referenced by these artifacts are associated with Secure Hash Algorithm with a 256-bit digest (SHA-256), allowing their integrity to be checked after export. The approach was evaluated across four releases using two publicly reproducible identity workloads. The reported results show that the prototype generates complete OSCAL packages with hash-linked evidence. Evidence gaps are represented through automatically generated findings and remediation items. The prototype also identifies evidence that falls outside the defined monitoring window. The authors distinguish compliance-as-code from policy enforcement alone, arguing that policy enforcement does not provide the traceability between regulatory obligations, enforcement points, and assessment evidence required for compliance assessment. Their approach therefore treats traceability, evidence completeness, provenance, and machine-readable assessment output as explicit parts of the compliance process.

Operational policy enforcement has also been studied through Policy-as-Code approaches in cloud-native software-delivery environments. For example, Nogueira and Resende [55] describe Policy-as-Code as an approach in which governance requirements are expressed as executable policies that can be versioned, tested, reviewed, and automatically enforced. Their study uses Open Policy Agent (OPA) [56] and its declarative Rego policy language to implement a shared policy-definition layer across two enforcement stages. During Continuous Integration, Kubernetes manifests are evaluated using Conftest, while the same Rego policies are applied at deployment time through OPA Gatekeeper and Kubernetes admission control. The approach was evaluated using 29 Kubernetes manifests, 37 experimental scenarios, eight resource types, and 261 policy assertions. Within this experimental setting, the shared policy definitions were reused across both stages, and intentionally introduced insecure configurations were detected without observing false positives or false negatives. The study therefore provides empirical evidence that reusable Policy-as-Code definitions can support consistent enforcement across complementary stages of a cloud-native software-delivery pipeline.

Automated security-policy checking has also been examined directly at the Infrastructure-as-Code level. Verdet et al. [57] study the implementation of security policies in Terraform configurations across Amazon Web Services (AWS), Azure, and Google Cloud. Their analysis uses Checkov [58] and Tfsec [59], two static analysis tools that provide predefined checks for evaluating the implementation of security policies in Infrastructure-as-Code configurations. The authors map security policies derived from recognized industry frameworks to corresponding checks supported by these tools and use them to analyze Terraform files from 812 open-source GitHub projects. The study shows how security-policy checks can be associated with concrete configuration patterns

in Terraform and evaluated automatically through static analysis. It also identifies an important boundary of this form of automation: not every compliance guideline can be translated into an identifiable configuration-level policy, and the implementation and enforcement of available policies remain uneven.

These studies illustrate two operational forms of automated policy assessment. Policy-as-Code enables governance requirements to be represented and enforced as executable policies across software-delivery stages, while static analysis tools such as Checkov and Tfsec evaluate Infrastructure-as-Code configurations through predefined security-policy checks. The methodology proposed in this thesis considers these forms of technical assessment within a broader evidence-grounded process that retains the regulatory requirement, standardized security-control context, assessment objective, selected system observations, evidence source, and resulting compliance decision as part of the assessment workflow.

4.4 AI-Driven Compliance Assessment

Recent studies have investigated the use of artificial intelligence to improve cybersecurity auditing and compliance assessment [60]. Traditional cybersecurity audits are highly dependent on manual activities for vulnerability identification, security control evaluation, and compliance verification, which makes them labor-intensive and time-consuming. As organizations continue to operate more and more complex digital infrastructures while complying with multiple regulatory requirements, conventional auditing approaches have become progressively more difficult to scale. Consequently, AI has been proposed as a means of automating vulnerability detection and supporting compliance assessments to improve both efficiency and accuracy.

Machine learning and natural language processing are identified as the two main AI technologies applied to cybersecurity auditing. ML models can learn patterns from historical security data to identify vulnerabilities and anomalies, while NLP techniques can analyze regulatory documents, organizational policies, and compliance checklists to support compliance verification. NLP can extract requirements concerning areas such as access control, data protection, and reporting obligations, and can support the comparison of organizational practices with regulatory requirements. These technologies enable organizations to adopt a more proactive approach to cybersecurity by accelerating audit activities and assisting compliance assessment processes.

Legal compliance automation has also been examined from the perspective of Requirements Engineering, where two main limitations in existing approaches are identified [61]. First, many previous methods analyze legal artifacts sentence by sentence, even though the interpretation of an individual sentence may depend on definitions, preceding statements, or cross-references located elsewhere in the document. Second, existing automation strategies generally focus on classification, information extraction, semantic similarity, or predefined rules, and do not provide explanations or justifications for compliance decisions. The

authors classify earlier approaches into three main strategies: extracting metadata or semantic frames and applying predefined compliance rules; comparing regulatory provisions through embeddings and semantic similarity; and using encoder-only transformer models such as Bidirectional Encoder Representations from Transformers (BERT) to classify which compliance rules are satisfied by a given text.

Each of these strategies is associated with specific limitations. Metadata and rule-based methods require substantial preparatory effort because the regulatory text must first be translated into a structured representation that can be processed automatically. For example, one approach defines a conceptual model comprising 56 GDPR-related metadata types and derives 23 completeness criteria that specify which combinations of information must be present in a privacy policy [62]. This process depends on a qualitative analysis of the regulation and repeated validation with legal experts, while transferring the methodology to another regulation or document type requires the corresponding conceptual model and completeness criteria to be developed again. The authors estimate that constructing these regulatory artifacts accounts for approximately 30–40% of the effort required to replicate their methodology, with the remaining effort devoted to developing the automated solution. Consequently, changes in regulatory requirements or the introduction of new concepts may require the underlying metadata structure, criteria, and associated processing rules to be revised.

Semantic-similarity-based methods reduce some of this dependence on manually specified matching rules, but provide a comparatively rough basis for compliance analysis. Semantic proximity between a regulatory requirement and a contractual statement does not necessarily correspond to compliance. In the GDPR case study presented in [63], some statements that satisfy a requirement receive low similarity scores because they express the required information through document-specific entities, whereas other statements obtain relatively high scores despite failing to satisfy the corresponding requirement. The problem becomes particularly important when a requirement contains several legally relevant elements: a statement may mention only some of these elements and therefore satisfy the requirement only partially, even though its overall wording remains semantically close to the regulatory text. Finally, classification-based approaches face a related limitation, since assigning a statement to a regulatory category indicates the presence of relevant content but does not establish whether every component of a multi-part requirement has been fulfilled.

To overcome these limitations, a compliance assessment framework based on generative LLMs is proposed. The framework analyzes regulatory text at the paragraph level using zero-shot prompting, allowing the model to evaluate compliance while generating textual explanations and justifications for its decisions. The approach was evaluated in Data Processing Agreements under the General Data Protection Regulation (GDPR). As a result, paragraph-level context consistently outperformed sentence-level analysis across multiple LLMs, and demonstrated the importance of preserving contextual information during compliance reasoning. However, the authors describe their evaluation as pre-

liminary and emphasize that further validation in different legal domains, as well as expert evaluation of the generated justifications, is required before the approach can be generalized.

Cybersecurity compliance is addressed from a different perspective by integrating AI, blockchain, and smart contracts for the enforcement of internal security policies and the dynamic adjustment of security controls [64]. In the proposed framework, security policies and compliance requirements are encoded as executable smart contracts. AI processes cyber threat intelligence, identifies threats and potential non-compliance, and informs decisions concerning security control adjustments. The blockchain is used as an immutable ledger for recording compliance checks, policy changes, AI decisions, and enforcement actions, thus providing an auditable record of system activities.

The framework uses NIST Cybersecurity Framework and ISO27001 policies and controls as references for defining internal compliance rules, while MITRE ATT&CK is used as a source of threat and mitigation information. Threat intelligence is processed using SecureBERT, and its contextual embeddings are passed to a Random Forest classifier. The classified threats are then cross-referenced with an internal policy database through a lightweight rule-based engine. A decision tree determines the required action, and the corresponding smart contract executes predefined measures such as disabling insecure protocols, changing access permissions, updating firewall rules, or isolating affected endpoints. Compliance checks and enforcement actions are recorded in Hyperledger Fabric and aggregated into compliance reports.

The framework was evaluated in a simulated enterprise environment, where it demonstrated improvements in both compliance enforcement effectiveness and response time compared with a human-centric process. Yet, the authors acknowledge that the framework mainly targets internal security policies and identify the integration of external regulations, along with policy ambiguity and blockchain interoperability, as key limitations and future research directions.

More recent studies have extended compliance automation beyond individual AI models toward Agentic AI systems composed of multiple specialized LLM-based agents. EUROCOMPLY is proposed as an agentic framework designed to assess datasets and AI/ML pipelines against the GDPR, the EU AI Act, and 3rd Generation Partnership Project (3GPP) standards throughout the Machine Learning Operations (MLOps) lifecycle [65]. The framework orchestrates multiple agents responsible for different compliance tasks and integrates Retrieval-Augmented Generation (RAG) to support regulatory reasoning. To improve the accuracy and traceability of compliance decisions, EUROCOMPLY employs a dual retrieval strategy that combines vector-based retrieval for direct regulatory queries with graph-based retrieval to capture relationships between regulatory dependencies. Retrieved regulatory information is incorporated into the compliance analysis to support the generation of structured, human-readable compliance reports while reducing hallucinations through retrieval by grounding the generated responses in a dedicated regulatory knowledge base.

This framework was evaluated using AI/ML telecommunications use cases and assessed through both human experts and an LLM-as-a-Judge evaluation

pipeline. The evaluation demonstrates the feasibility of applying Agentic AI to the assessment of regulatory compliance. However, disagreements between human experts and automated evaluation have also been observed, particularly for complex compliance reports.

In summary, the literature demonstrates a clear evolution from traditional AI-assisted compliance automation toward LLM-based reasoning. Nevertheless, these approaches focus mainly on legal regulations, organizational policies, or AI governance frameworks, while comparatively limited attention has been dedicated to cybersecurity compliance assessment based on structured security control catalogs, evidence collected from real system artifacts, and transparent evidence-driven reasoning.

Chapter 5

Methodological Approach

This chapter presents the methodology developed for cybersecurity compliance assessment. The proposed approach starts from cybersecurity requirements defined by a directive or regulatory source and connects them with standardized security controls and observable system evidence. Relevant system evidence is associated with individual requirements to form grounded assessment cases, which can be evaluated using different decision mechanisms. In addition to the final compliance decision, the methodology considers the traceability and use of supporting evidence, allowing the assessment process to be examined beyond decision agreement alone. The chapter concludes by defining the measures used to evaluate the individual stages of the approach and the structured representation of the resulting assessment.

5.1 Cybersecurity Compliance Assessment

The proposed methodology defines a structured approach to cybersecurity compliance assessment by connecting regulatory requirements with evidence obtained from the system under assessment. The main objective of this methodology is to preserve a traceable relationship between the original regulatory requirements, the evidence available in the assessed system, and the final compliance decision. In fact, rather than evaluating regulatory requirements directly from their textual descriptions, the methodology associates these requirements with a standardized security-control context and relevant system observations. The resulting grounded information is then evaluated using different decision mechanisms.

The process begins with cybersecurity requirements extracted from the directive under assessment. To provide a common context for the subsequent assessment stages, these requirements are mapped to controls obtained from a standardized security-control catalogue. Two mapping approaches are considered: a reference mapping defined by the NATO experts and an automated mapping generated by a language model.

After the mapping stage, each regulatory requirement is connected with evidence describing the state of the system under assessment. The evidence is obtained from system sources such as configuration data, infrastructure state, and runtime information. Relevant properties extracted from these sources are stored as structured system observations together with information identifying their origin. This allows the evidence used in an assessment to be traced back to the system source from which it was obtained. Evidence grounding determines which of the available observations are relevant to a regulatory requirement and its associated security-control context.

The evidence-grounding stage supports two observation-selection strategies:

1. **Manual observation selection:** Manually defined rules associate each regulatory requirement with relevant system observations. These rules provide the reference baseline for observation selection.
2. **Language-model-based observation selection:** A language model selects system observations considered relevant to the regulatory requirement and its associated security-control context.

The two strategies provide alternative ways of selecting evidence from the same set of available system observations. Their selections can therefore be compared to examine the extent to which the automated approach agrees with the manually defined reference.

The regulatory requirement, its mapped security-control context, and the selected system observations form a grounded assessment case that is processed by rule-based and language-model-based evaluators. The rule-based evaluator applies explicit conditions to the available observations and serves as a deterministic baseline. The language-model-based approach evaluates the requirement in relation to its security-control context and the available evidence. In addition to the compliance decision, the language-model-based evaluator can provide an explanation of the assessment and indicate the evidence considered in reaching the decision. Both evaluators use the same decision space i.e., **PASS**, **FAIL**, and **UNKNOWN**, which enables their outputs to be compared under a common assessment setting.

An important aspect of the methodological approach is that the evaluation does not only consider the correctness of the final compliance decision. In fact, a correct decision alone does not show whether the supporting evidence has been used appropriately. The methodological approach therefore also considers the traceability of evidence and its consistency with the resulting assessment. For language-model-based assessments, explanations and remediation suggestions can also be examined independently from the predicted compliance state.

Figure 5.1 summarizes the proposed methodology together with its execution flow. As can be seen, the assessment process is subdivided into three main stages: security-control mapping, evidence grounding, and compliance evaluation. This separation supports independent comparison of expert and language-model-generated mappings, manual and language-model-based observation selection, and rule-based and LLM-based compliance decisions.

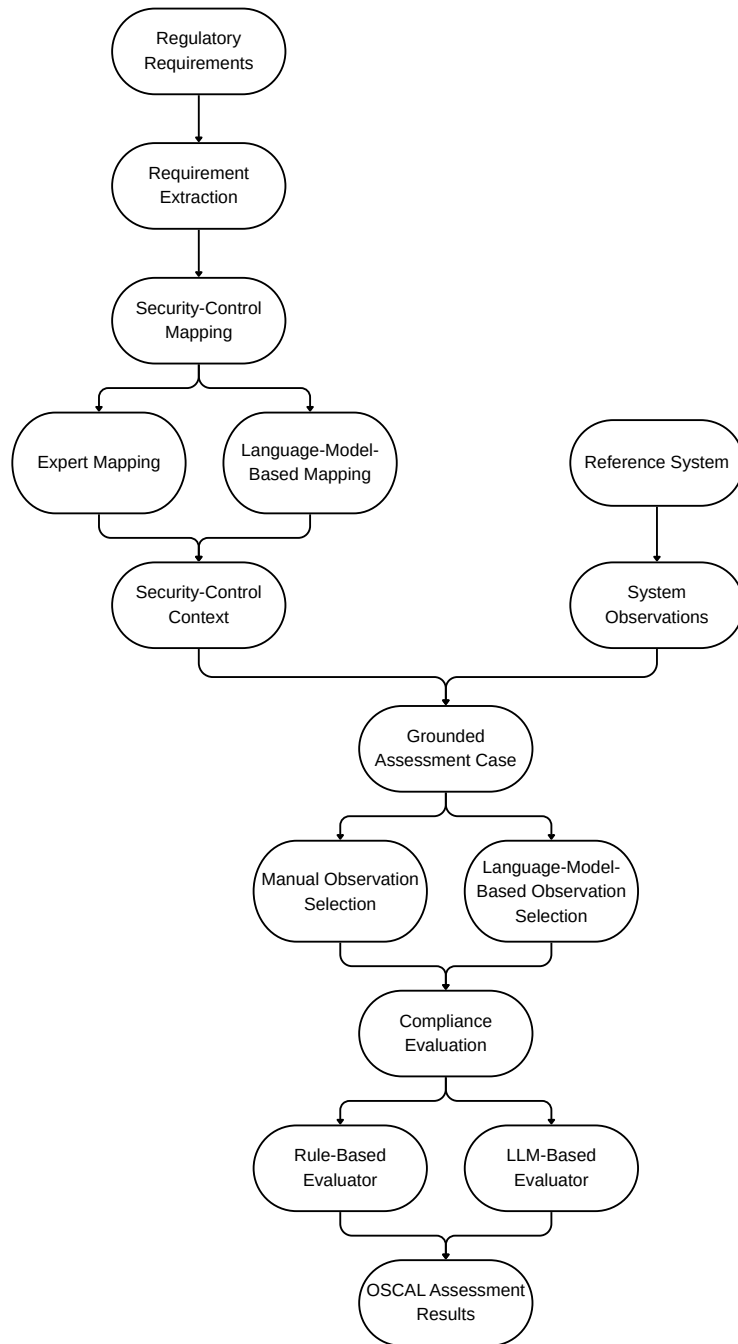


Figure 5.1: Main components and execution flow of the proposed methodology

In what follows, the various components are described in detail with the support of a running example concerning a regulatory requirement stating that communication with an Internet-facing service must be protected in transit. The requirement can be associated with a security control concerned with communication protection. Evidence collection may provide a system observation indicating whether transport encryption is enabled for the service, and evidence grounding identifies this observation relevant to the requirement and its security-control context. Compliance evaluation then uses the observed system state to determine whether the available evidence supports the requirement. This running example is used throughout the following sections to illustrate the main stages of the methodology.

5.2 Security-Control Mapping

The security-control mapping stage establishes a connection between regulatory requirements and controls from a standardized security-control catalogue. The purpose of this stage is to provide a common security-control context for each regulatory requirement before system evidence is associated with it. Regulatory sources and security-control catalogues may organize and express their requirements at different levels of detail. Therefore, the methodology does not assume a one-to-one correspondence between regulatory requirements and security controls.

In the running example, the requirement concerning protection of Internet-facing communication is associated with a security control addressing communication protection. The regulatory requirement expresses the obligation to be assessed, while the mapped security control provides a standardized security context for interpreting that obligation.

Before the mapping stage, the regulatory requirements are extracted from the source directive and prepared for use in the assessment process. The original identifier and description of each requirement are preserved so that the assessment remains traceable to the corresponding requirement in the regulatory source.

The expert-defined and language-model-based mappings are maintained separately so that the automated mapping can be evaluated against the expert-defined reference without treating model-generated mappings as authoritative.

5.2.1 Expert Mapping

The expert mapping establishes a reference association between regulatory requirements and controls from a standardized security-control catalogue. Each regulatory requirement is associated with one or more security controls that represent its relevant security context. If the catalogue distinguishes between base controls and more specific control enhancements, this distinction is preserved in the mapping.

The expert mapping serves as the reference against which automated mapping can be evaluated. For each regulatory requirement, the mapped security controls provide the security-control context used in the subsequent assessment stages. The regulatory requirement and its associated security-control context are therefore maintained as the assessment proceeds to evidence grounding.

5.2.2 Language-Model-Based Mapping

The second mapping approach uses a language model to associate each regulatory requirement with controls from a standardized security-control catalogue. The mapping task is formulated as a closed-list selection problem rather than an unrestricted search over the complete catalogue. For each regulatory requirement, the model receives the requirement identifier and description together with a predefined set of candidate security controls. Each candidate is identified by its control identifier and title, and the model selects the controls that it considers most relevant to the regulatory requirement.

The candidate set is determined according to the category associated with each regulatory requirement. Categories may be identified, for example, through prefixes in the requirement identifiers, with a separate set of candidate security controls defined for each category. Where no category-specific set is available, a generic candidate set can be used.

In the running example, the communication-protection requirement may be presented to the language model together with a candidate set containing security controls related to communication protection, access control, and system configuration. The model selects the control or controls most relevant to the requirement.

Generated mappings are validated before they are accepted. Selections outside the permitted candidate set and duplicate identifiers are removed, while outputs that do not contain a usable mapping are rejected. This process checks the validity of the generated selection. The accepted language model mappings are maintained separately from the expert mappings and are evaluated by comparison with the expert reference.

5.3 Evidence Grounding

The evidence-grounding stage connects each regulatory requirement with evidence obtained from the system under assessment. System evidence is represented through observations describing security-relevant properties of the assessed system. From the available observations, those relevant to a particular regulatory requirement and its associated security-control context are selected through either manually defined rules or a language-model-based selection approach. The regulatory requirement, its security-control context, and the selected observations together form a grounded assessment case. Information about the source of each selected observation is retained so that the evidence used in the assessment remains traceable to the assessed system.

In the running example, the evidence collection produces an observation describing whether transport encryption is enabled for the Internet-facing service. Evidence grounding associates this observation with the communication-protection requirement because the observed property is relevant to determining whether communication is protected in transit. The requirement, its security-control context, and this observation then form the basis of the grounded assessment case.

The separation between evidence collection and observation selection is intentional. Evidence collection obtains security-relevant system properties and represents them as system observations. Observation selection determines which of the available observations are relevant to a particular regulatory requirement and its associated security-control context. Compliance evaluation is performed only after the relevant evidence has been selected and the grounded assessment case has been formed.

5.3.1 System Evidence Collection

System evidence is collected from identifiable sources associated with the system under assessment. These sources may provide information about system configuration, infrastructure state, runtime state, or other security properties. Relevant properties obtained during collection are represented as system observations.

Each observation retains information identifying the source from which it is obtained. This allows the evidence used in later assessment stages to be traced back to the assessed system. The collection process establishes the set of available observations, from which relevant evidence is subsequently selected for individual regulatory requirements.

In the running example, evidence collection may obtain information about the configuration of the Internet-facing service and represent the relevant property as a system observation indicating whether transport encryption is enabled. The observation also retains information about the evidence source from which this property was obtained.

Each system observation describes the state of a security-relevant property of the system under assessment. If the required system property cannot be observed, the evidence is treated as unavailable. This distinction prevents missing evidence from being interpreted automatically as evidence of non-compliance.

5.3.2 Observation Selection

Once the system observations have been collected, the grounding process determines which observations are relevant to each regulatory requirement. The methodology supports two observation-selection strategies: manually defined observation rules and language-model-based observation selection. Both strategies operate on the same set of available system observations established during evidence collection.

Manual observation rules define which system observations are relevant to a regulatory requirement and its associated security-control context. Each rule associates a regulatory requirement with one or more observations that provide evidence for its assessment. Where assessment objectives are available, they help identify the specific aspect of the associated security control addressed by the selected evidence. An assessment objective describes a specific aspect of a security control that can be examined during assessment. The manually defined rules provide the reference observation selection used to evaluate the language-model-based selection approach.

The second observation-selection strategy uses a language model to identify the system observations relevant to a regulatory requirement. The model considers the regulatory requirement together with its associated security-control context and the set of available system observations. The model identifies the most relevant observations from the available set for assessing the regulatory requirement.

In the running example, a manual observation rule may associate the requirement for protected communication with the observation describing whether transport encryption is enabled. The rule therefore identifies this observation as relevant evidence for assessing the requirement. For the same requirement, the language-model-based approach performs this association automatically. Given the communication-protection requirement, its security-control context, and the available observations, the model may select the transport-encryption observation as relevant evidence.

After observation selection, a grounded assessment case is formed. It combines the regulatory requirement, its associated security-control context, and the selected system observations. If assessment objectives are used, they identify the specific aspects of the associated security controls addressed by the selected evidence.

5.4 Compliance Evaluation

Each grounded assessment case is evaluated to determine whether the available system evidence supports the corresponding regulatory requirement. Evidence grounding and compliance evaluation serve different purposes. Evidence grounding identifies the system observations relevant to a regulatory requirement and connects them with its associated security-control context. Compliance evaluation interprets this grounded information and produces a compliance decision.

Two approaches are considered for compliance evaluation: a deterministic rule-based evaluator and an LLM-based evaluator. Both operate on grounded assessment cases but differ in how they interpret the available evidence. The rule-based evaluator applies predefined decision criteria, whereas the LLM-based evaluator interprets the regulatory requirement, security-control context, and grounded evidence to determine a compliance decision.

5.4.1 Rule-Based Evaluation

The rule-based evaluator applies predefined decision criteria to the system observations included in a grounded assessment case. The same criteria are applied consistently across assessment cases, making the resulting decisions deterministic and reproducible. The relationship between the regulatory requirement, its security-control context, and the relevant system observations is established during evidence grounding. Therefore, the rule-based evaluation focuses on interpreting the state represented by the selected evidence.

The rule-based evaluator combines the selected observations according to predefined decision criteria. These criteria distinguish evidence that supports the requirement, evidence that does not support it, and cases in which the available evidence is insufficient to determine a decision.

In the running example, a predefined rule may determine how the observed state of transport encryption contributes to the assessment of the communication-protection requirement. Since the same decision criteria are applied each time, the same grounded evidence produces the same assessment result.

5.4.2 LLM-Based Evaluation

The LLM-based approach evaluates the regulatory requirement using its associated security-control context and the evidence contained in the grounded assessment case. The security-control context supports interpretation of the requirement, while the selected system observations provide information about the state of the system under assessment. Based on this information, the model produces a compliance decision and can provide an explanation and remediation guidance associated with that decision.

A distinction is maintained between contextual information and system evidence. The security-control context supports the interpretation of the regulatory requirement, whereas the selected system observations represent the evidence obtained from the system under assessment. The compliance decision is therefore grounded in the selected observations rather than inferred from the security-control context alone.

For the running example, the LLM-based approach interprets the communication protection requirement together with its security-control context and the selected transport-encryption observation. The model uses the observation as system evidence when producing its compliance decision and may also explain how that evidence relates to the requirement.

For LLM-based assessments, the evidence referenced in the generated output can be compared with the evidence contained in the grounded assessment case. This makes it possible to examine whether the reported reasoning is based on the evidence available to the model rather than considering only the final compliance decision.

The two evaluation approaches support different forms of analysis. The rule-based evaluator provides a reproducible evidence-to-decision procedure, while the LLM-based evaluator can additionally produce explanations and remedia-

tion guidance. Therefore, assessment quality can be examined not only through the compliance decision but also through the model’s use of the grounded evidence.

5.5 Evaluation Metrics

The proposed methodology is evaluated at several stages rather than only at the final compliance decision since errors may arise during security-control mapping, observation selection, or compliance evaluation. Separate measures are defined to examine security-control mapping, observation selection, decision agreement, evidence traceability, evidence use, and remediation output.

Compliance Decision Evaluation

The primary decision-level metric measures agreement between the evaluator output and the reference label associated with the grounded assessment case. For a control i , an agreement is recorded when the evaluator decision is identical to the corresponding reference label:

$$A_i = \begin{cases} 1, & \text{if } \hat{y}_i = y_i, \\ 0, & \text{otherwise,} \end{cases} \quad (5.1)$$

where $\hat{y}_i$ denotes the evaluator decision and y_i denotes the reference label. The overall reference agreement is then calculated as:

$$\text{Reference Agreement Rate} = \frac{\sum_{i=1}^N A_i}{N}, \quad (5.2)$$

where N is the total number of evaluated assessment cases.

Mapping Evaluation

The language-model-generated security-control mappings are compared with the expert-defined reference mappings at several levels. An exact-set comparison alone may be too restrictive when a regulatory requirement can be associated with multiple security controls or when a security-control catalogue distinguishes between base controls and more specific enhancements. Four comparison levels are therefore considered:

- exact match,
- direct overlap,
- base-control overlap, and
- family-level overlap.

An *exact match* occurs when the language-model-based mapping contains exactly the same security controls as the expert-defined reference mapping. The order of the selected controls is not considered. If one or more controls are missing or additional controls are selected, the mapping is not considered an exact match.

A less restrictive comparison is the *direct overlap*, which occurs when the language-model-based mapping and the expert-defined reference mapping share at least one identical security control.

The *base-control overlap* normalizes control enhancements to their corresponding base controls before comparison. Finally, *family-level overlap* compares the broader control categories to which the mapped security controls belong. This provides a less restrictive indication of whether the automated and reference mappings place a regulatory requirement within the same general security area.

For example, two mappings may select different security-control identifiers while still referring to the same underlying base control or to controls within the same security family. The different comparison levels distinguish these cases from mappings that have no common security-control context.

Observation Selection Evaluation

Observation-selection measures how closely the observations selected by the language model correspond to those defined in the manual observation rules. For each regulatory requirement, similarity between the manually selected observation set O_M and the language-model-selected observation set O_S is calculated using the Jaccard similarity coefficient:

$$J(O_M, O_S) = \frac{|O_M \cap O_S|}{|O_M \cup O_S|}. \quad (5.3)$$

A value of 1 indicates that both approaches selected the same observation set, whereas a value of 0 indicates that they selected no common observations. For example, if the manual approach selects three observations and the language model selects three observations, with two observations shared by both selections, their intersection contains two observations and their union contains four. The resulting Jaccard similarity is therefore $2/4 = 0.5$.

This metric measures agreement between the two observation selections rather than establishing the correctness of either selection. A low Jaccard value therefore indicates that the approaches rely on different system observations, but does not by itself determine which selection is more appropriate for the regulatory requirement.

Jaccard similarity is calculated for each regulatory requirement, and the overall score $\bar{J}$ is obtained by averaging the individual similarity values:

$$\bar{J} = \frac{1}{N} \sum_{i=1}^N J_i. \quad (5.4)$$

Evidence Traceability and Faithfulness

Reference agreement alone does not indicate whether an evaluator used the grounded system evidence correctly. For this reason, the methodology also examines traceability and evidence faithfulness. Table 5.1 summarizes the evaluation metrics.

Table 5.1: Metrics for evidence traceability and faithfulness

Metric	What it Measures
Traceability rate	Whether grounded evidence fields remain linked to the assessment objectives associated with the selected security controls
Decision-evidence consistency rate	Whether the evaluator decision agrees with the decision directly implied by the grounded evidence
Used-evidence validity rate	Whether evidence identifiers reported by the evaluator exist in the grounded evidence supplied to that case
Decisive-evidence coverage rate	Whether the evaluator reports the evidence fields responsible for a FAIL or UNKNOWN result
Explanation contradiction rate	Whether the generated explanation contains statements that conflict with the values of the grounded evidence

The *traceability rate* is computed at the evidence-field level. For each assessment case, grounded evidence fields are compared with the evidence-field identifiers referenced by the selected assessment objectives. The rate is calculated as the number of matching evidence fields divided by the total number of grounded evidence fields considered by the metric.

The *decision-evidence consistency rate* is computed over the assessment cases containing grounded evidence. The evidence-derived decision is determined using a fixed precedence: the presence of any false observation results in **FAIL**; the presence of an unavailable observation results in **UNKNOWN**; otherwise, the presence of a true observation results in **PASS**. The rate is the number of cases in which the evaluator decision matches this evidence-derived decision divided by the number of non-empty assessment cases.

The *used-evidence validity rate* is computed over the evidence identifiers reported in the evaluator’s **used_evidence** output. Each reported identifier is compared with the grounded evidence fields supplied in the same assessment case. The rate is calculated as the number of matching identifiers divided by the total number of reported evidence identifiers.

The *decisive-evidence coverage rate* is computed for the assessment cases containing decisive evidence. If one or more observations are false, all false observations are treated as decisive. If no false observation is present, unavailable observations are treated as decisive. For each applicable case, the proportion of

decisive evidence fields included in `used_evidence` is calculated. The final rate is the average of these per-case proportions.

The *explanation contradiction rate* is computed over the non-empty assessment cases using a deterministic pattern-based procedure. Predefined textual aliases for evidence fields are located in the explanation, and the surrounding text is checked for patterns inconsistent with the corresponding grounded evidence value. The rate is calculated as the number of cases containing at least one detected contradiction divided by the number of non-empty assessment cases.

Remediation Evaluation

Remediation output is evaluated using both end-to-end and conditional measures. The evaluation considers five aspects, namely: whether remediation is provided when it is required by the reference decision, whether remediation is generated when the evaluator produces a **FAIL** decision, whether the generated remediation relates to the failing evidence and provides a specific recommendation, and whether remediation is unnecessarily produced when corrective action is not required. Table 5.2 summarizes these measures.

Table 5.2: Metrics used to evaluate remediation output

Metric	What it Measures
Remediation coverage rate	Whether remediation is provided for assessment cases with a FAIL reference decision
Conditional remediation presence rate	Whether remediation is provided when the evaluator produces a FAIL decision
Conditional remediation relevance rate	Whether remediation generated following an evaluator FAIL decision addresses the failing evidence
Conditional remediation specificity rate	Whether remediation generated following an evaluator FAIL decision provides a sufficiently specific recommendation rather than a generic response
Unnecessary remediation rate	Whether remediation is provided for assessment cases with a PASS reference decision

The *remediation coverage rate* is computed over assessment cases with a **FAIL** reference decision. The rate is calculated as the number of these cases containing non-empty remediation text divided by the total number of cases with a **FAIL** reference decision. As an end-to-end measure, the rate may be affected by disagreement at the compliance-decision stage.

The *conditional remediation presence rate* is computed over cases in which the evaluator produces a **FAIL** decision. It is calculated as the number of these cases containing non-empty remediation text divided by the total number of evaluator-produced **FAIL** decisions.

The *conditional remediation relevance rate* is computed over cases in which the evaluator produces a **FAIL** decision and remediation is present. For each case, the remediation text is checked for a textual reference to at least one false evidence field using predefined aliases associated with the available evidence fields. The rate is the number of cases satisfying this condition divided by the number of evaluator-produced **FAIL** cases with remediation.

The *conditional remediation specificity rate* uses the same set of evaluator-produced **FAIL** cases with remediation. Remediation is counted as specific when it contains at least five words and does not contain any of the predefined generic remediation phrases. The rate is calculated as the number of cases satisfying these conditions divided by the number of evaluator-produced **FAIL** cases with remediation.

The *unnecessary remediation rate* is computed over assessment cases with a **PASS** reference decision. The rate is calculated as the number of these cases containing non-empty remediation text divided by the total number of cases with a **PASS** reference decision. Cases with an **UNKNOWN** reference decision are excluded from both the remediation coverage and unnecessary remediation rates, since insufficient or unavailable evidence does not establish whether corrective remediation is required.

These remediation measures are automated heuristic indicators intended to provide repeatable checks of basic presence, evidence relevance, and specificity properties within the proof-of-concept evaluation.

Referring to the running example, when the assessment identifies insufficient protection of communication in transit, relevant remediation should address the issue directly. A specific recommendation should also indicate a concrete corrective action rather than provide only a generic instruction to review the system configuration.

OSCAL Output Validation

The generated assessment-results representation is validated against the OSCAL assessment-results schema. This validation checks whether the generated assessment results conform to the structure and data types defined by the OSCAL schema. This check verifies properties such as the expected document structure, permitted fields, and data types. A successful validation confirms that the generated representation conforms to the OSCAL schema.

Chapter 6

Experimental Setup

This chapter describes the environment and configurations used to evaluate the proposed methodological framework by presenting the framework implementation, the reference system, the experimental dataset, the model configuration, and the combinations of mapping, observation-selection, and evaluation methods used in the experiments.

6.1 Framework Implementation

The experimental framework is implemented as a set of Python components organized around the main orchestration script, `pipeline.py`. The script provides the entry point for an experimental run and coordinates the loading of the required inputs, the generation of grounded assessment cases, the compliance evaluation, the metric computation, and the OSCAL export.

The implementation separates the main functions of the framework into dedicated modules. Regulatory-text parsing and extraction are implemented under the `data` module, while system-evidence collection and grounding are implemented under `system_evidence`. The two compliance evaluators are implemented under `evaluators`, and OSCAL result generation is handled separately under `oscal`. NIST SP 800-53 assessment context is retrieved through the `NISTCatalog` component from the local OSCAL representation of the NIST catalogue.

The selected D/0048 requirements, D/0048-to-NIST mappings, observation-selection rules, and collected system-evidence snapshots are prepared before the assessment run and supplied to the pipeline as structured inputs.

A run is configured through command-line arguments specifying, among others, the mapping file, observation-rule file, system-evidence file, evaluator mode, and output run name.

Table 6.1: Functionalities of the main software components of the proposed experimental framework

Component	Functionalities
<code>pipeline.py</code>	Orchestrates assessment-case generation, executes evaluator, computes metrics, and optional OSCAL export.
<code>system_evidence/collect_evidence.py</code>	Collects and normalizes evidence from the reference system.
<code>system_evidence/generator.py</code>	Constructs grounded assessment cases from mappings, observation rules, and collected system evidence.
<code>nist_catalog.py</code>	Retrieves NIST SP 800-53 control and assessment-objective context from the local OSCAL catalogue.
<code>evaluators/rule_based.py</code>	Implements deterministic compliance evaluation.
<code>evaluators/llm_based.py</code>	Provides the language-model-based compliance evaluator interface.
<code>evaluators/metrics.py</code>	Computes evaluation and traceability-related metrics from assessment results.
<code>oscal/ export_assessment_results.py</code>	Transforms internal assessment results into OSCAL Assessment Results format.

Assessment-Case Generation and Grounding

The central integration point between the regulatory, security-control, and system-evidence layers is the **SystemEvidenceGenerator**. For each D/0048 requirement, this component first loads the corresponding mapping record and obtains the assessment context for the mapped NIST controls through **NISTCatalog**. The assessment context includes the NIST control information and the assessment objectives used by the observation-selection rules.

Observation-selection rules are stored as structured JSON records. Each rule identifies a system observation together with the NIST control and an assessment-objective text fragment to which the observation is associated. During grounding, the generator resolves this objective against the assessment context produced from the currently selected mapping. It then resolves the selected observation identifier against the collected system-evidence snapshot.

The resulting grounded assessment case is represented as a structured record containing the D/0048 requirement, its mapping, assessment context, grounded evidence, evidence provenance, system snapshot information, and an evidence-derived reference decision. The evidence supplied to the evaluator is represented as a compact dictionary that associates observation identifiers with their values. Provenance is retained separately for each selected observation and includes its collection status, source-file reference, collection timestamp, and raw value.

Listing 6.1 shows an excerpt from the grounding implementation. For each selected observation identifier, the generator retrieves the corresponding collected observation, derives the value supplied to the evaluator, and retains the associated provenance information. If the observation is missing or its collection status is not successful, the grounded value is represented as **None**, which is serialized as **null** in the corresponding JSON artifacts.

Listing 6.1: Grounding selected system observations into evidence with provenance

```
for observation_rule in rule.get("observations", []):
    observation_id = observation_rule["id"]
    observation = self.observations.get(observation_id)

    if observation is None:
        value = None
        collection_status = "missing"
        source = None
        collected_at = self.system_evidence.get("metadata",
        {}).get("collected_at")
        raw_value = None
    else:
        value = observation.get("value")
        collection_status = observation.get(
        "collection_status", "error")
        source = observation.get("source")
```

```

        collected_at = observation.get("collected_at")
        raw_value = observation.get("raw_value")

    evidence[observation_id] = (
        value if collection_status == "success" else None
    )
    provenance[observation_id] = {
        "collection_status": collection_status,
        "source": source,
        "collected_at": collected_at,
        "raw_value": raw_value,
    }

```

Evaluation and Output Flow

After the grounded assessment cases have been generated, `pipeline.py` selects the compliance evaluator according to the run configuration. The evaluator interface receives a representation of the regulatory requirement containing its mapping and assessment context together with the grounded evidence. This common interface allows the same generated assessment cases to be processed by either evaluator implementation.

The rule-based evaluator uses the grounded evidence directly to apply the deterministic decision procedure described in Section 6.5. The language-model-based evaluator uses the same evidence together with the regulatory requirement, mapping, and assessment context when constructing the model input. The resulting evaluation is then attached to the corresponding assessment case.

Listing 6.2 shows the implementation in the main pipeline. The evaluator receives the same grounded evidence representation independently of the selected evaluation method. The returned evaluation is then stored together with the mapping, assessment context, evidence, provenance, and evidence-derived reference decision.

Listing 6.2: Evaluator invocation and assessment-result construction in the pipeline

```

evaluation_control = {
    **control,
    "mapping": case.get("mapping"),
    "assessment_context": case.get("assessment_context"),
}
evaluation = evaluator.evaluate(evaluation_control,
case["evidence"])
results.append({
    "control": control,
    "mapping": case.get("mapping"),
    "assessment_context": case.get("assessment_context"),

```

```

    "evidence": case["evidence"],
    "evidence_provenance": case.
    get("evidence_provenance"),
    "system_snapshot": case.get("system_snapshot"),
    "expected": case.get("expected"),
    "evaluation": evaluation,
    "match": evaluation.get("decision") == case.
    get("expected"),
  })

```

Listing 6.2 shows how the information required to trace the evaluator output back to the grounded assessment case is preserved. The `match` field records whether the evaluator decision agrees with the evidence-derived reference decision; the corresponding aggregate measure is therefore treated as reference agreement.

For each run, the generated assessment cases are first stored in `generated_dataset.json`. After evaluation, the complete results are written to an evaluator-specific `evaluation_results.json` file. The same result objects are subsequently passed to the metric component, which produces the `metrics.json` file. The result objects are also passed to the OSCAL exporter to generate the `assessment- results.json` file.

Run outputs are isolated using separate run directories. In the experiment, separate directories are retained for each combination of the five reference system states and four experimental configurations, resulting in 20 separately stored sets of experimental outputs.

6.2 Reference System

The experiments are conducted on a controlled reference system implemented locally using Terraform and Docker that provides the environment from which the system observations used in the compliance assessment are collected. The reference system consists of an application service, a database, an Nginx reverse proxy, Docker networks, and persistent storage. The choice of a local reference system is motivated by its ability to make the system configuration and runtime state directly available for inspection during the experiments. The reference system configuration described in this section is used as the baseline experimental state, denoted as `S0` in what follows. Four additional states (`S1--S4`) are derived from `S0` through isolated changes to selected security-relevant properties, as described in Section 6.4.

System Architecture

The reference system follows a container-based architecture composed of three main services: a Flask application [66], a PostgreSQL database [67], and an Nginx reverse proxy [68]. The services run in separate Docker [69] containers and communicate through Docker networks. Docker is therefore used both to

isolate the individual services and to define how communication between them takes place.

The Flask application represents the application service under assessment. It runs inside the `compliance_app` container and communicates with the PostgreSQL database through an internal Docker network. PostgreSQL provides persistent application storage through a dedicated Docker volume. In `S0`, the application itself is not exposed directly on the host. Instead, external requests pass through the Nginx reverse proxy, which forwards them to the Flask application.

Two Docker networks are defined in the reference system. The `compliance_internal` network connects Nginx, the application, and the database, while Nginx is also connected to the `compliance_external` network. This makes Nginx the externally accessible entry point, while the application and database are not directly exposed.

Figure 6.1 illustrates the main components of the reference system and their communication paths.

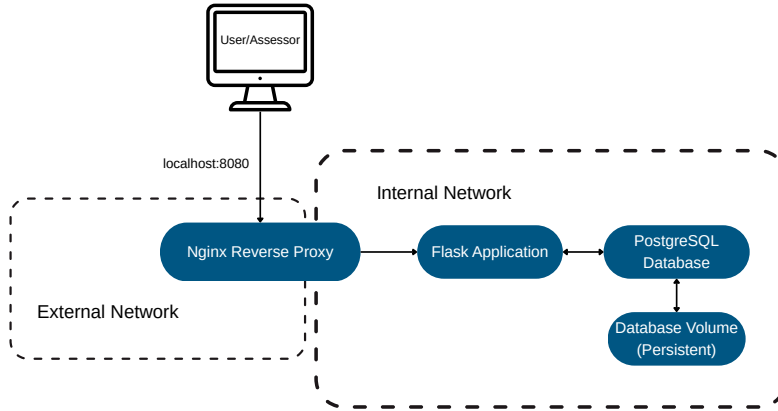


Figure 6.1: Architecture of the reference system in state `S0`

The infrastructure itself is defined using Terraform [70], an infrastructure-as-code tool that represents infrastructure resources through declarative configuration files. In this implementation, Terraform describes the Docker resources required by the reference system, including the containers, networks, and persistent volume. The Docker provider allows these Docker resources to be represented as part of the Terraform configuration. In addition to making the experimental infrastructure explicit, this representation provides infrastructure state and configuration information that can be inspected during evidence collection.

Table 6.2 summarizes the technologies used in the reference system and their roles in the experiment.

Table 6.2: Main technologies and components of the reference system

Technology	Component	Role in the Reference System
Terraform	Infrastructure definition	Defines and manages the Docker containers, networks, and persistent volume used by the reference system.
Docker	Container runtime	Provides isolated runtime environments for the application, database, and reverse proxy.
Flask	<code>compliance_app</code>	Provides the application service under assessment and exposes the application health endpoint.
PostgreSQL	<code>compliance_database</code>	Provides persistent database storage for the application.
Nginx	<code>compliance_nginx</code>	Acts as the externally accessible reverse proxy and forwards requests to the application container.
Docker networks	<code>compliance_internal</code> , <code>compliance_external</code>	Separates internal service communication from the network used by the externally accessible reverse proxy.
Docker volume	Database volume	Provides persistent storage for the PostgreSQL container.

Implementation and Runtime Configuration

The application image is built locally as `compliance-app:latest`. Inside the container, the Flask application runs on port 5000. Nginx listens on port 80 inside its container, which is published on the host as port 8080. Requests received by the reverse proxy are forwarded to the Flask application at `http://compliance_app:5000`. The application is therefore accessed through Nginx rather than by exposing the Flask container directly.

The application also provides a health endpoint that returns a simple health-status response. This endpoint is exposed through Nginx at `http://localhost:8080/health` and is used during evidence collection to determine whether the application is reachable.

In `S0`, the reverse proxy is exposed through HTTP on port 80 and no TLS-enabled listener is configured. The corresponding system observation is therefore `tls_configured=false`. TLS is enabled separately in state `S1` as one of the controlled modifications of the baseline system.

The main runtime settings are summarized in Table 6.3.

Table 6.3: Runtime configuration of the reference system in state `S0`

Configuration Item	Setting
Application image	<code>compliance-app:latest</code>
Application container port	5000
Application runtime user	<code>appuser</code> (non-root)
Database image	<code>postgres:16-alpine</code>
Reverse-proxy image	<code>nginx:1.27-alpine</code>
Nginx container port	80
Host-exposed port	8080
Nginx upstream	<code>http://compliance_app:5000</code>
Health endpoint	<code>http://localhost:8080/health</code>
Internal network	<code>compliance_internal</code>
External network	<code>compliance_external</code>
Infrastructure provider	<code>kreuzwerker/docker</code> (~3.0)

6.3 Experimental Dataset

The system observations are derived from several sources associated with the reference system. These include Terraform state and plan information, Docker container and network state, Nginx configuration, application health endpoint, and compliance-related artifact files stored with the application configuration.

Terraform information is obtained from the structured output of the Terraform state and plan commands. Docker runtime information is collected by inspecting the application, database, and reverse-proxy containers together with

the configured networks. The Nginx configuration is read directly from the reference system configuration, while application availability is checked through the health endpoint. Additional compliance-related properties are read from JSON artifact files maintained within the reference system. Source snapshots are retained before the individual system observations are derived from them.

The experimental dataset consists of 30 regulatory requirements selected from NATO D/0048 [71] and 57 system observations. The requirements were selected based on their applicability to the technical properties observable in the containerized reference system and through the available evidence sources. Consequently, the selected set focuses on requirements that can be examined using machine-collected technical evidence. The selected regulatory requirements cover eight security measure groups: Protection of Services (POS), Identity and Access Management (IAM), Configuration Management (CM), Logging, Continuous Monitoring and Audit (LMA), Incident Response (IR), Continuity Planning (CP), Protection of Software (PSW), and Data Protection (DA). Their distribution is presented in Table 6.4.

Table 6.4: Distribution of D/0048 regulatory requirements used in the experiment

Prefix	D/0048 Security Measure Group	Requirements
POS	Protection of Services	6
IAM	Identity and Access Management	7
CM	Configuration Management	5
LMA	Logging, Continuous Monitoring and Audit	5
IR	Incident Response	2
CP	Continuity Planning	2
PSW	Protection of Software	2
DA	Data Protection	1
Total		30

The same set of 57 observation identifiers is used across the five reference system states. When a system property is successfully determined, its observation value is represented as `true` or `false`. A value of `true` indicates that the observed property is present or enabled, whereas `false` indicates that the property was observed to be absent or disabled. When the corresponding evidence is unavailable, the observation value is represented as `null`.

The values of the observations vary according to the state of the reference system. In each state, 54 observations have a successful collection status and three observations have the status `not_collected`. The resulting distribution of observation values is shown in Table 6.5.

The three observations represented as `null` are `application_health_endpoint_reachable`, `runtime_matches_terraform_configuration`, and `baseline_verification_matches_runtime`. These observations remain unavailable in all five reference system states. Their `null` values originate from the evidence-

Table 6.5: Distribution of system-observation values across reference system states

State	true	false	null	Total
S0	48	6	3	57
S1	49	5	3	57
S2	47	7	3	57
S3	47	7	3	57
S4	46	8	3	57

collection process. They are preserved during grounding and they are not introduced by manually modifying the generated assessment cases.

The remaining differences in observation values result from the controlled changes made to the reference system. These state-specific changes are described in Section 6.4.

6.4 Controlled Reference System States

As already stated, the experiments consider five controlled states of the reference system, defined as S0--S4. State S0 corresponds to the reference system configuration described in Section 6.2. Each of the S1--S4 states introduces an isolated change to a selected security-relevant property of S0. The states are not cumulative; each modified state is derived independently from S0 while the remaining experimental variables retain their S0 values.

The states are defined through state-specific Terraform variable files. These files modify the deployed reference system configuration rather than directly modifying collected observations or compliance decisions. After a state is applied, the standard evidence-collection process is executed against the resulting system. The resulting state-specific evidence snapshot is then used as input to the assessment pipeline. This design allows changes in the deployed system to propagate through evidence collection and grounding to the resulting compliance decision.

Table 6.6 summarizes the five states and the security-relevant changes introduced in each case.

The state definitions do not directly assign values to these observations. Instead, the corresponding system properties are changed in the deployed reference system and subsequently observed by the evidence collector. For example, enabling TLS in S1 changes the collected value of `tls_configured` from `false` in S0 to `true` in S1. Similarly, S2, S3, and S4 introduce changes in network exposure, application runtime privileges, and restart-policy configuration, respectively.

Within each reference system state, the same state-specific evidence snapshot is used by all four experimental configurations. Consequently, differences between Configurations A–D within a state result from the mapping, observation-

Table 6.6: Controlled reference system states used in the experiments

State	Description	Configuration Change	Affected Observation(s)
S0	Baseline	Reference system configuration described in Section 6.2	None
S1	TLS enabled	TLS is enabled at the Nginx reverse proxy	<code>tls_configured</code>
S2	Application publicly exposed	The application is directly exposed outside the internal network	<code>application_not_publicly_exposed</code>
S3	Application running as root	The application container is configured to run as root	<code>application_runs_as_non_root</code>
S4	No restart policy	The application restart policy is removed	<code>application_restart_policy_configured</code> , <code>all_containers_restart_policy_configured</code>

selection, or evaluation method rather than from differences in the underlying collected system state.

6.5 Model Configuration

The language-model-based components of the experiments are implemented using `Qwen/Qwen2.5-1.5B-Instruct`. The same model was used for three tasks in the experimental pipeline:

- mapping D/0048 regulatory requirements to relevant NIST SP 800-53 security controls,
- generation of observation selection rules, and
- language-model-based compliance evaluation.

A single model is used across these stages to keep the model choice fixed throughout the proof-of-concept implementation. The model was selected as a relatively small open-weight model that can be executed locally, allowing the language-model-based stages to operate without sending regulatory information or system evidence to an external model service.

The model is loaded locally using the Hugging Face Transformers library [72]. The mapping and observation-selection tasks use the Transformers text-generation pipeline with automatic device placement. For compliance evaluation, the model and tokenizer are loaded through `AutoTokenizer` and `AutoModelForCausalLM`. The compliance evaluator is executed on the CPU using `torch.float32`.

Deterministic generation is used in all language-model-based stages by setting `do_sample=False`, which disables stochastic token sampling. The output length is not dynamically tuned during the experiments; instead, a fixed `max_new_tokens` value is assigned to each task based on the expected structured JSON output. Mapping uses 180 tokens, since it returns one to three NIST identifiers and a short rationale. Observation selection uses 220 tokens, since it can return up to three observation-objective links. Compliance evaluation uses 320 tokens, since the response includes a decision, used evidence, explanation, and remediation. These values are kept constant across all final runs.

Table 6.7 summarizes the model configuration used for the three model-based tasks.

Table 6.7: Configuration of the model-based tasks using Qwen2.5-1.5B-Instruct

Task	Maximum New Tokens	Sampling
NIST control mapping	180	Disabled
Observation selection	220	Disabled
Compliance evaluation	320	Disabled

For NIST control mapping, the model receives a D/0048 requirement together with a restricted set of candidate NIST controls. It selects the controls considered relevant to the requirement and returns the corresponding control identifiers.

For observation selection, the model receives the mapped assessment context together with the identifiers of the available system observations. Its task is to identify the observations that are relevant to the assessment.

For compliance evaluation, the model receives the grounded assessment case, including the regulatory requirement, its security-control context, and the selected system evidence. It produces a compliance decision together with an explanation, the evidence used, and remediation guidance where applicable.

The mapping and observation-selection tasks are implemented using the Transformers text-generation pipeline with automatic device placement. The compliance evaluator loads the same model directly by `AutoModelForCausalLM` and is configured for CPU execution.

The rule-based evaluator applies the same decision procedure to all grounded assessment cases. Its decision is derived from the values of the selected system observations, which may be `true`, `false`, or `null`. Table 6.8 summarizes the implemented decision logic.

A `false` observation takes precedence over unavailable evidence and therefore results in a `FAIL` decision even if another selected observation has the value `null`. If no `false` value is present but at least one selected observation is `null`, the evaluator returns `UNKNOWN`. If no grounded evidence is available, the decision is also `UNKNOWN`. A `PASS` decision is produced only when all selected evidence values are `true`. Requirement-specific thresholds, weights, or exceptions are not applied, in order to keep the rule-based evaluation deterministic and consistent

Table 6.8: Decision logic implemented by the rule-based evaluator

Evidence Condition	Decision
No grounded evidence is available	UNKNOWN
At least one selected evidence value is false	FAIL
No false value exists, but at least one selected evidence value is null	UNKNOWN
All selected evidence values are true	PASS

across the assessed requirements.

For the language-model-based stages, separate prompt templates are used for security-control mapping, observation selection, and compliance evaluation. The same prompt templates are used across all five reference-system states. The complete prompt templates are provided in Appendix A.

6.6 Experimental Configurations

Four configurations are used in the experiments. They differ in the source of the D/0048-to-NIST mapping, the method used to select system observations, and the compliance evaluator. All four configurations assess the same 30 D/0048 regulatory requirements. Within each reference system state, the four configurations use the same state-specific system-evidence snapshot.

The four combinations used in the experiments are shown in Table 6.9. For clarity, the four experimental configurations are referred to as Configurations A–D throughout the remainder of the thesis.

Table 6.9: Configurations used in the experiments

Configuration	Mapping	Observation Selection	Evaluator
A	Expert	Manual	Rule-based
B	Expert	Manual	Language-model-based
C	Model-generated	Model-generated	Rule-based
D	Model-generated	Model-generated	Language-model-based

Configurations A and B use the expert-defined D/0048-to-NIST mapping and the manually defined observation rules. They differ only in the compliance evaluator. Configurations C and D use the D/0048-to-NIST mapping and observation rules generated by the language model.

Each of the four assessment configurations is executed against each of the five reference-system states described in Section 6.4, resulting in 20 experimen-

tal runs. Within a given state, the collected system-evidence snapshot is held constant across Configurations A–D, while the mapping, observation-selection, and evaluation components vary according to the configuration.

6.7 Output and Validation

Each experimental run produces an assessment dataset containing the grounded assessment cases and the corresponding compliance-evaluation outputs. For each D/0048 requirement, the stored result includes the associated NIST mapping, the assessment context, the selected system evidence and its provenance, the reference decision, and the decision produced by the evaluator. The outputs required for computing the evaluation measures are stored separately from the assessment data.

In addition to the experiment-specific output, each run produces an OSCAL assessment-results document. Each document contains the 30 assessment cases included in the corresponding experimental run.

OSCAL Assessment-Results Representation

The OSCAL output represents the main elements of the assessment using the assessment results model. The NIST controls associated with the D/0048 requirements are included as reviewed controls. Grounded system evidence is represented through OSCAL observations, with the observed value stored as a property of the corresponding observation. Information about the source of each observation is also retained in the OSCAL output, preserving the link between the assessment result and the underlying system evidence.

For each assessed D/0048 requirement, the OSCAL output contains a finding representing the result of that assessment. The D/0048 requirement identifier is retained as a property of the corresponding assessment records, and the compliance decision is represented through the status associated with the assessed NIST control. **PASS** decisions are represented as **satisfied**, while **FAIL** decisions are represented as **not-satisfied**. Decisions for which compliance cannot be determined are also represented as **not-satisfied** with explicit properties indicating that the internal assessment decision is **UNKNOWN** and that the result is undetermined due to unavailable or insufficient evidence. Explanations, remediation information, and undetermined-case remarks produced during compliance evaluation are included in the finding remarks when available.

The generated document uses the OSCAL structures required to represent the assessment results, including reviewed controls, observations, findings, finding targets, relevant evidence, and back-matter resources.

Schema Validation

The generated OSCAL representation is checked against the OSCAL assessment results schema `oscal_assessment-results.schema.json`, version 1.2.2.

This validation checks whether the generated JSON document conforms to the structure and data constraints defined by the OSCAL schema.

Chapter 7

Experimental Results

This chapter presents the experimental results obtained from the evaluation of the proposed proof-of-concept framework. The evaluation is intended to examine the behaviour and feasibility of the proposed assessment approach under the defined experimental conditions. Notably, the chapter examines the effects of the controlled reference system states on the resulting assessments, compares the rule-based and language-model-based compliance evaluators, and reports the results for security-control mapping and observation selection. An evaluation of evidence traceability, faithfulness, and the remediation guidance produced by the language-model-based evaluator is also presented.

7.1 Effects of the Controlled Reference System States

The first set of experiments examines how the controlled changes introduced in the reference system are reflected in the collected system observations and in the resulting compliance assessments. This evaluation is important because an evidence-grounded assessment framework should remain sensitive to relevant changes in the actual state of the assessed system. By introducing isolated changes to selected security-relevant properties, it is possible to trace whether each change is captured by the evidence-collection process and whether it propagates through grounding to the corresponding assessment results.

As described in Section 6.4, five reference system states are considered. **S0** represents the baseline configuration, while **S1--S4** introduce an isolated modification to one security-relevant property of **S0**. The states are not cumulative. Consequently, the results for each modified state can be compared with the baseline to identify the effects of the corresponding system change.

It is important to recall that the same 57 system-observation identifiers are retained across the five reference system states. In each state, 54 observations are successfully collected and three remain unavailable (see Table 6.5). The

distribution of **true** and **false** values changes according to the controlled modification introduced in each state.

These changes in observation values correspond to the intended modifications of the reference system. In **S1** state, enabling TLS changes **tls_configured** to **true**. In **S2** state, directly exposing the application sets **application_not_publicly_exposed** to **false**. Similarly, in **S3** state, configuring the application container to run as root sets **application_runs_as_non_root** to **false**. Finally, removing the restart policy in **S4** state sets **application_restart_policy_configured** and **all_containers_restart_policy_configured** to **false**. The three unavailable observations remain represented as **null** in all five states.

To examine how these changes affect the assessment results, the expert mapping and manually defined observation-selection rules are kept fixed across the five reference system states. Therefore, Configurations A and B use the same grounded assessment cases, differing only in the compliance evaluator applied to them. Table 7.1 presents the distribution of the evidence-derived reference decisions for this path.

Table 7.1: Evidence-derived reference decisions for the expert mapping and manually defined observation selection assessment path

State	PASS	FAIL	UNKNOWN	Total
S0	21	6	3	30
S1	22	5	3	30
S2	19	8	3	30
S3	19	8	3	30
S4	20	7	3	30

It is important to note that the changes in the reference-decision distributions can be traced to specific requirements and observations. Table 7.2 reports the requirements whose evidence-derived reference decision changes with respect to the **S0** baseline.

Table 7.2: Reference-decision changes caused by the controlled system modifications with respect to the baseline

State	Requirement Label	Baseline State Decision	Modified-State Decision
S1	POS-A	FAIL	PASS
S2	POS-B	PASS	FAIL
	POS-C	PASS	FAIL
S3	IAM-A	PASS	FAIL
	IAM-B	PASS	FAIL
S4	POS-D	PASS	FAIL

In **S1**, enabling TLS produces the only observed transition from **FAIL** to

PASS. Specifically, requirement POS-A, which concerns the use of TLS, is affected by the change in `tls_configured` from `false` to `true`. Its reference decision therefore changes from FAIL to PASS, increasing the number of PASS reference decisions from 21 to 22.

In S2, the change in application exposure affects requirements POS-B and POS-C. Both requirements use `application_not_publicly_exposed` as selected evidence. When this observation changes from `true` to `false`, both reference decisions change from PASS to FAIL, resulting in 19 PASS and 8 FAIL reference decisions.

In S3, running the application container as root affects requirements IAM-A and IAM-B, concerning privileged-account use and the principle of least privilege. For both requirements, `application_runs_as_non_root` is selected as evidence and changes from `true` to `false`. Consequently, both reference decisions change from PASS to FAIL.

In S4, removal of the restart policy affects requirement POS-D, which concerns the management of virtual machines and containers throughout their lifecycle. The selected observation `all_containers_restart_policy_configured` changes from `true` to `false`, causing the reference decision to change from PASS to FAIL. This results in 20 PASS and 7 FAIL reference decisions.

The three UNKNOWN reference decisions remain unchanged across all five states. They correspond to requirements CM-A, CM-B, and LMA-A. For CM-A, the observation `runtime_matches_terraform_configuration` remains unavailable, while CM-B is associated with the unavailable observations `baseline_verification_matches_runtime` and `runtime_matches_terraform_configuration`. For LMA-A, `application_health_endpoint_reachable` remains unavailable. Since these observations remain unavailable throughout the five reference system states, the corresponding assessment cases remain UNKNOWN.

Across the five states, the expert/manual assessment path produces 150 requirement-state assessment cases: 101 PASS, 34 FAIL, and 15 UNKNOWN. These cases are not treated as 150 statistically independent samples, since the same 30 regulatory requirements are evaluated repeatedly across five controlled states of the same reference system. The additional states are useful for the proof-of-concept because they allow the assessment process to be exercised under different system conditions rather than only against a single configuration. In particular, the controlled modifications make it possible to verify whether changes in security-relevant system properties are captured by the collected observations and propagated through the grounding process to the corresponding compliance decisions. They therefore provide evidence that the proposed framework can respond to changes in the assessed system while preserving the relationship between the system state, the supporting evidence, and the resulting decision.

7.2 Rule-Based and Language-Model-Based Evaluation

This section examines the decisions produced by the rule-based and language-model-based evaluators. Evaluating their behaviour is important because the final compliance decision is the outcome of the mapping, observation-selection, and evidence-grounding stages. The comparison therefore examines whether the resulting evaluator decisions remain consistent with the decisions supported by the grounded evidence.

The analysis considers the four experimental configurations introduced in Section 6.6. Configurations A and C use the rule based evaluator, whereas Configurations B and D use the language-model-based evaluator. For each configuration, the same 30 regulatory requirements are evaluated across the five controlled reference system states, resulting in 150 requirement-state assessment cases per configuration. For each assessment case, a reference decision is derived from the grounded evidence using the deterministic decision rules defined in Section 5.4.1. Specifically, the reference decision is **FAIL** when at least one selected observation is **false**, **UNKNOWN** when no observation is **false** and at least one is unavailable, and **PASS** when all selected observations are **true**. Reference agreement, as defined in Section 5.5, then indicates whether the decision produced by the evaluator matches this evidence-derived reference decision. Table 7.3 summarizes the resulting evaluator decisions and reference agreement across the five reference system states.

Table 7.3: Aggregate compliance evaluation results across the five reference system states

Config.	Reference Decisions	Evaluator Decisions	Reference Agreement
A	101 / 34 / 15	101 / 34 / 15	100%
B	101 / 34 / 15	136 / 14 / 0	76.67%
C	116 / 34 / 0	116 / 34 / 0	100%
D	116 / 34 / 0	143 / 7 / 0	82%

Decision distributions are reported in the order **PASS** / **FAIL** / **UNKNOWN**.

As expected, the rule-based evaluator achieves complete reference agreement in both Configurations A and C. In each configuration, all 30 assessment cases agree with their corresponding reference decisions in every system state, resulting in an aggregate reference agreement of 100%.

This result should be interpreted in relation to how the reference decisions are constructed. The evidence-derived reference decisions and the rule-based evaluator use the same deterministic decision semantics for interpreting the selected observations. Consequently, complete agreement is expected when the same grounded evidence is processed correctly. The results therefore provide a consistency check of the implemented evidence-to-decision path rather than an

independent estimate of compliance classification performance.

The different reference-decision distributions in Configurations A and C result from their different grounding processes. Configuration A uses the expert security-control mapping and manual observation selection, whereas Configuration C uses model-generated mapping and observation selection.

The evaluator decision distributions in Table 7.3 show a tendency towards PASS decisions in both language-model-based configurations. In Configuration B, 136 of 150 evaluator decisions are PASS, compared with 101 PASS reference decisions. In Configuration D, 143 evaluator decisions are PASS, compared with 116 PASS reference decisions. Neither language-model-based configuration produces an UNKNOWN decision. In particular, Configuration B contains 15 reference UNKNOWN cases, all of which are classified as PASS by the language-model-based evaluator.

Table 7.4 shows the breakdown of the reference agreements of the two language-model-based configurations for each controlled system state. The

Table 7.4: Reference agreements of the language-model-based evaluator across system states

State	Configuration B	Configuration D
S0	80%	80%
S1	80%	83.33%
S2	73.33%	86.67%
S3	73.33%	80%
S4	76.67%	80%
Aggregate	76.67%	82%

language-model-based evaluator does not reproduce the evidence-derived reference decisions as consistently as the deterministic evaluator. Configuration B obtains an aggregate reference agreement of 76.67%, with 115 of 150 assessment cases agreeing with the reference decisions. Configuration D obtains an aggregate reference agreement of 82%, with 123 of 150 cases in agreement.

Figure 7.1 provides a visual comparison of the per-state reference agreements of Configurations B and D. As can be seen, Configuration D has a higher reference-agreement rate than Configuration B in four of the five reference system states, while both configurations obtain the same agreement in S0. The different security-control mappings and different observation selections used by the two configurations produce different grounded assessment cases and different evidence-derived reference distributions. In Configuration B, the reference distribution consists of 101 PASS, 34 FAIL, and 15 UNKNOWN decisions, whereas Configuration D has no reference UNKNOWN decisions. Therefore, the difference in reference agreement cannot be attributed to a single experimental component.

This behaviour is also visible in the aggregate confusion matrices shown in Tables 7.5 and 7.6.

For both configurations, all reference PASS cases are also classified as PASS

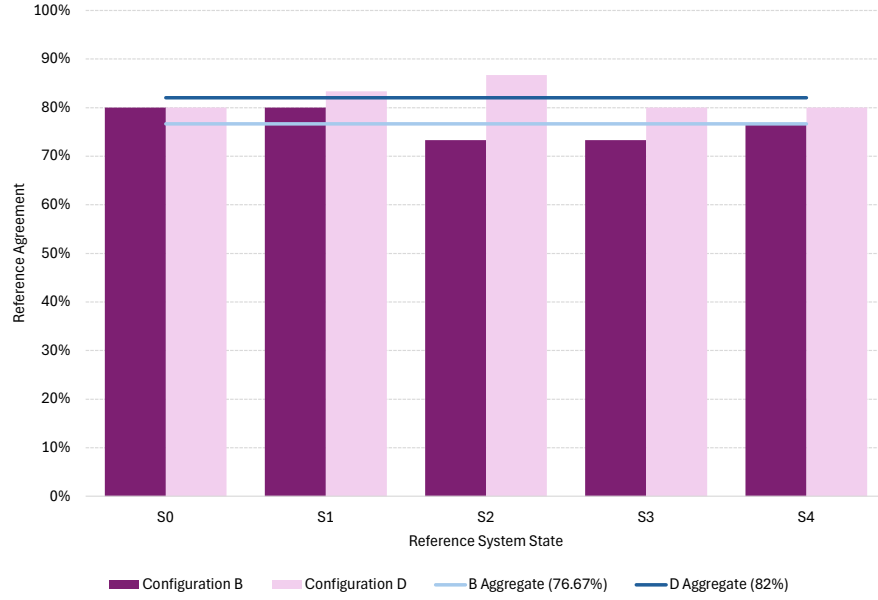


Figure 7.1: Reference agreement of the language-model-based evaluator per reference system state

Table 7.5: Aggregate confusion matrix for Configuration B

	Evaluator Decision			
		PASS	FAIL	UNKNOWN
	PASS	101	0	0
	FAIL	20	14	0
	UNKNOWN	15	0	0

Table 7.6: Aggregate confusion matrix for Configuration D

	Evaluator Decision			
		PASS	FAIL	UNKNOWN
	PASS	116	0	0
	FAIL	27	7	0
	UNKNOWN	0	0	0

by the language-model-based evaluator. The disagreements occur when the reference decision is **FAIL** or **UNKNOWN**. In Configuration B, 20 of the 34 reference **FAIL** cases are classified as **PASS**. In Configuration D, this number is even larger, that is, 27. In addition, all the 15 reference **UNKNOWN** cases of Configuration B are classified as **PASS**.

These results indicate that, under the evaluated conditions and selected model, disagreement is not distributed uniformly across the three decision states. The selected language model reproduces the evidence-derived **PASS** decisions but more frequently returns **PASS** when the reference decision indicates negative or insufficient evidence.

Behaviour under Unavailable Evidence

As already stated, unavailable evidence is represented separately from an observed negative system condition in the proposed compliance assessment framework. Within the expert/manual assessment path, unavailable evidence contributes to an **UNKNOWN** reference decision when no selected observation already implies **FAIL**. As a result, three regulatory requirements receive an **UNKNOWN** reference decision in each of the five system states because their selected evidence includes unavailable observations and no selected observation implies **FAIL**. Since the same three requirements remain affected across all five states, this results in 15 **UNKNOWN** requirement–state cases in total.

In Configuration B, the grounded assessment cases produced using the expert mapping and manual observation selection are evaluated by the language-model-based evaluator. For the 15 cases whose reference decision is **UNKNOWN**, the evaluator returns **PASS** in every case and does not produce any **UNKNOWN** decision. These cases correspond to three regulatory requirements repeated across the five system states, that is, **CM-A**, concerning the development and maintenance of baseline configurations; **CM-B**, concerning the verification of system configurations against the approved baseline; and **LMA-A** concerning continuous monitoring for abnormalities, attacks, and unauthorized use. Thus, within the evaluated expert/manual grounding path, the model does not preserve the distinction between insufficient evidence and evidence supporting compliance.

Configuration D does not provide an equivalent test of this behaviour. Its model-generated observation-selection rules do not select the three observations that remain unavailable across the reference system states. Consequently, none of the grounded assessment cases in Configuration D contains unavailable selected evidence, and no evidence-derived **UNKNOWN** reference decisions are produced. This result shows that the handling of unavailable evidence depends not only on the compliance evaluator, but also on the preceding observation-selection stage, which determines whether unavailable observations become part of the grounded assessment case.

Within the scope of the proof of concept, the results from Configuration B demonstrate the importance of representing unavailable evidence explicitly in the language-model-based compliance evaluation. A compliance assessment process needs to distinguish between evidence that indicates that a requirement is

satisfied and evidence that is unavailable or insufficient to reach a decision. The observed behaviour demonstrates the importance of preserving this distinction and provides a clear direction for improving the handling of unavailable evidence in language-model-based evaluation.

7.3 Security-Control Mapping Evaluation

The security-control mapping determines the standardized security-control context used in the subsequent stages of the assessment. Evaluating this stage is important for examining how closely the model-generated mappings correspond to the expert-defined mappings before they are used for evidence grounding.

The model-generated security-control mappings are compared with the expert-defined mappings using the comparison levels defined in Section 5.5, that is, exact mapping match, direct overlap, base-control overlap, and family-level overlap. Exact match requires the complete mappings to match, while direct overlap measures shared security-control identifiers. Base-control overlap ignores control enhancements, and family-level overlap compares mappings at the broader security-family level. Table 7.7 reports the agreement and overlap obtained at each comparison level for the 30 regulatory requirements. As can be noticed,

Table 7.7: Comparison of model-generated and expert-defined security-control mappings

Comparison Level	Cases	Rate
Exact match	0/30	0%
Direct overlap	16/30	53%
Base-control overlap	20/30	67%
Family-level overlap	24/30	80%

no model-generated mapping exactly matches the complete set of security controls in the corresponding expert-defined mapping. However, direct overlap is observed for 16 out of the 30 requirements, indicating that at least one identical security control is selected by both approaches in slightly more than half of the cases.

The level of agreement increases when the comparison is performed at broader levels of the security-control hierarchy. After control enhancements are normalized to their corresponding base controls, overlap is observed in two thirds of the cases. At the security-family level, this percentage increases to 80%. This means that 24 of the 30 mappings share at least one common family.

These results show that the model-generated mappings differ substantially from the expert-defined mappings at the level of exact control selection, while greater similarity is observed when the comparison is performed at the base control and security-family levels.

7.4 Observation-Selection Evaluation

Observation-selection determines which system observations are included in the grounded assessment case for each regulatory requirement. Evaluating this stage is important, since differences in the selected observations directly affect the evidence available to the subsequent compliance evaluator. The comparison examines how closely the model-generated observation selections correspond to the manually defined selections.

For each of the 30 regulatory requirements, the two approaches produce a set of selected observation identifiers. Since these sets can contain different numbers of observations, their agreement cannot be represented only by counting identical selections. As defined in Section 5.5, Jaccard similarity measures the proportion of observations shared by the two sets relative to the observations selected by either approach. A value of 1 indicates identical observation sets, while a value of 0 indicates that the two sets have no observations in common.

The comparison shows that no regulatory requirement has an identical observation set under the two selection approaches. In contrast to the security-control mapping stage, the manually defined and model-generated sets have no observations in common for 16 of the 30 requirements. The remaining 14 requirements contain at least one observation selected by both approaches, although their complete observation sets differ. Across all 30 requirements, the average Jaccard similarity is 16%.

These results indicate limited overlap between the model-generated observation selections and the manually defined reference. More importantly, the differences occur at the evidence-selection stage of the assessment process. As a result, the two approaches can provide different evidence to the compliance evaluator even when they assess the same regulatory requirement. This observation is relevant when interpreting the compliance evaluation results, since differences in evaluator decisions may reflect not only the evaluator itself, but also the grounded evidence produced by the preceding observation-selection stage.

7.5 Evidence Traceability and Faithfulness

Evaluating evidence traceability and faithfulness is important because a compliance decision should not only be produced, but should also remain connected to the system evidence on which the assessment is based. These properties make it possible to examine whether the generated decision and explanation are consistent with that evidence.

The four experimental configurations are evaluated using the measures defined in Section 5.5. The evaluation considers whether evidence provenance is preserved, whether the evidence referenced by the evaluator is valid, whether the decision is reflected in the evaluator output, and whether the generated explanation contradicts the evidence. Table 7.8 reports the aggregate values of these five measures across the five reference system states for Configurations A–D.

Table 7.8: Aggregate evidence traceability and faithfulness metrics

Metric	A	B	C	D
Traceability rate	100%	100%	100%	100%
Used-evidence validity	100%	100%	100%	100%
Decision-evidence consistency	100%	77%	100%	82%
Decisive-evidence coverage	100%	63%	100%	56%
Explanation-contradiction	0%	7%	0%	11%

The traceability rate is 100% for all four configurations. This means that the generated assessment cases preserve the structural links between the assessment context, selected system observations, and their evidence sources. In particular, the observations used in an assessment remain associated with their source information, allowing the evidence supporting an assessment case to be traced back to the corresponding system artifact or evidence source. As discussed in Section 5.5, this result concerns the completeness of the implemented traceability structure. It confirms that the framework preserves the required evidence links across all evaluated configurations.

Used-evidence validity is also 100% across all configurations. This indicates that every observation identified by an evaluator as supporting its decision is present in the corresponding grounded assessment case. In other words, the evaluators do not refer to evidence identifiers that are absent from the evidence provided for the assessment.

The faithfulness-related measures distinguish the rule-based and language-model-based configurations more clearly. Configurations A and C, which use the rule-based evaluator, obtain 100% for both decision-evidence consistency and decisive-evidence coverage, with an explanation-contradiction rate of 0%. This was expected due to the construction of the rule-based evaluator. Its decision and explanation are generated directly from predefined rules applied to the grounded evidence.

In contrast, the language-model-based configurations, i.e., B and D, show a different behaviour. Decision-evidence consistency is 77% for Configuration B and 82% for Configuration D, indicating that some model decisions differ from the decision implied by the grounded evidence under the predefined decision semantics. Decisive-evidence coverage is 63% and 56%, respectively, indicating that the evidence reported by the model does not always include all evidence fields considered relevant for the corresponding assessment. Finally, the explanation-contradiction rate is 7% for Configuration B and 11% for Configuration D, which means that some generated explanations contain statements that conflict with the grounded evidence values.

Taken together, these results show three distinct behaviours in the evaluated language-model outputs:

- disagreement with the evidence-derived decision,

- incomplete reporting of decisive evidence, and
- occasional contradictions between the generated explanation and the grounded evidence.

7.6 Evaluation of Remediation Coverage and Relevance

Remediation guidance is evaluated because a compliance-assessment result should not only identify a problem, but can also provide useful information to address it. Evaluating the generated remediation therefore helps determine whether the language-model-based evaluator provides actionable guidance when the assessment does not result in **PASS**.

The remediation outputs of Configurations B and D are evaluated using the measures defined in Section 5.5. The evaluation considers overall remediation coverage, the presence of remediation when necessary, its relevance and specificity, and whether remediation is generated when it is not needed. Table 7.9 reports these five measures for Configurations B and D across the five reference system states.

Table 7.9: Aggregate remediation evaluation metrics for the language-model-based configurations

Metric	Configuration B	Configuration D
Coverage	41%	47%
Conditional presence	100%	100%
Conditional relevance	64%	100%
Conditional specificity	100%	100%
Unnecessary remediation	0%	0%

The end-to-end remediation coverage is 41% for Configuration B and 47% for Configuration D. These values are closely related to the compliance decisions reported in Section 7.2. In Configuration B, 20 of the 34 reference-**FAIL** cases are classified as **PASS**, while in Configuration D, 27 of the 34 reference-**FAIL** cases are classified as **PASS**. Consequently, remediation coverage is affected by the compliance decision itself: when a reference-**FAIL** case is classified as **PASS**, the evaluator does not identify the case as requiring remediation.

This distinction becomes clearer when remediation is examined only for cases that the evaluator classifies as **FAIL**. Remediation is generated for all 14 evaluator-**FAIL** cases in Configuration B and all 7 evaluator-**FAIL** cases in Configuration D. Therefore, once the evaluator identifies a case as **FAIL**, remediation is consistently provided. The lower end-to-end coverage mainly results from reference-**FAIL** cases being classified as **PASS**, rather than from a failure to generate remediation after an evaluator-**FAIL** decision.

For the conditional remediation relevance rate, the relevance heuristic is satisfied in 9 of 14 cases in Configuration B and all 7 cases in Configuration D. The conditional remediation specificity is 100% for both configurations, as the specificity heuristic is satisfied for all cases where remediation is provided following a **FAIL** decision.

Another inconsistency is observed in Configuration D. In 9 reference-**FAIL** cases classified as **PASS** by the evaluator, the model still generated remediation guidance. In these cases, the generated output identifies a need for corrective action without reflecting it in the final compliance decision. The implementation preserves the generated fields as returned by the model and does not enforce consistency between the decision and remediation fields.

The unnecessary remediation rate is 0% in both configurations, as no remediation is generated for cases with a **PASS** reference decision.

Overall, the remediation evaluation shows that the language-model-based evaluator consistently generates remediation when it identifies a case as **FAIL**. The lower end-to-end coverage is mainly associated with reference-**FAIL** cases that are classified as **PASS**, while the generated remediation is generally specific and, particularly in Configuration D, relevant to the identified issue. The observed decision-remediation inconsistency in Configuration D also shows the importance of evaluating remediation together with the compliance decision rather than as an isolated output. Within the proof of concept, these results demonstrate that remediation can provide useful corrective guidance, while its overall effectiveness remains closely linked to the preceding compliance decision.

Chapter 8

Conclusion

This thesis addressed the problem of cybersecurity compliance assessment, with particular attention to the difficulty of connecting high-level regulatory requirements with concrete properties of the system being assessed. This problem is particularly relevant in cybersecurity because compliance decisions often depend on technical information distributed across system configurations, infrastructure definitions, runtime state, and other system artifacts.

To address this problem, the thesis proposed a methodology for evidence-grounded and traceable cybersecurity compliance assessment. Rather than evaluating regulatory text in isolation, the proposed approach connects regulatory requirements with a standardized security-control context and system observations collected from the assessed environment.

The proposed methodology supports security-control mapping, observation selection, evidence grounding, compliance evaluation, and machine-readable result generation. Both deterministic and language-model-based alternatives were considered for selected stages of the assessment process. The resulting assessment cases preserve connections between the regulatory requirement, the security-control context, the selected system observations, their evidence sources, and the final compliance decision.

To assess the proposed methodology, a proof of concept was developed and evaluated through a set of controlled experiments. The experimental evaluation used 30 regulatory requirements and a containerized reference system under five controlled system states. The states introduced isolated changes to security-relevant system properties, allowing their effects to be followed through evidence collection, grounding, and compliance evaluation. Four experimental configurations were used to compare expert and model-generated security-control mappings, manual and model-generated observation selections, and rule-based and language-model-based compliance evaluation.

The experiments showed the feasibility of the proposed methodological approach and provided insight into its behaviour under the evaluated conditions. Notably, the implemented methodology preserved a traceable connection between changes in the assessed system and the resulting compliance assessment.

The controlled system modifications produced the expected changes in the corresponding system observations, and these changes propagated to the evidence-derived reference decisions when the affected observations were relevant to a regulatory requirement.

The comparison of security-control mappings and observation selections also showed that the model-generated outputs can differ substantially from the expert-defined and manually defined references. Mapping agreement increased when evaluated at broader levels of the security-control hierarchy, while the model-generated observation selections showed limited overlap with the manually defined selections.

For compliance evaluation, the deterministic evaluator reproduced the evidence derived reference decisions by construction. The language-model-based evaluator showed lower reference agreement and a tendency to return **PASS** for cases with negative or unavailable evidence. In particular, all reference-UNKNOWN cases, corresponding to three regulatory requirements evaluated across the five system states, were classified as **PASS** through the expert/manual grounding path. This result demonstrates the importance of preserving evidence availability throughout the assessment process and of examining how language-model-based evaluators distinguish unavailable evidence from observed evidence.

These results should be interpreted within the experimental design used in this thesis work. The mapping and observation-selection methods are paired in the experimental configurations, so evaluator behaviour is compared within a common grounding path. In addition, the reference decisions are derived from the grounded evidence using deterministic decision semantics rather than from independent expert judgement. The reported reference agreement therefore characterizes consistency with the evidence-derived reference decisions rather than independent compliance-classification accuracy.

The faithfulness and remediation measures identified some limitations in the language-model outputs, including incomplete reporting of decisive evidence, occasional explanation contradictions, and inconsistencies between compliance decisions and remediation guidance.

Despite these limitations, the usage of language models is a promising research direction for further development and evaluation. Future work could evaluate the proposed approach using different language models and model sizes. The framework used a single model configuration, and extending the experiments to other models would provide a broader understanding of how model choice influences the different language-model-based stages of the assessment process.

The framework could also be applied to a wider range of regulatory requirements and system environments. The same methodology could be extended to additional D/0048 requirements and to systems with more complex infrastructure, identity, networking, and deployment characteristics. This would also make it possible to study to what extent different classes of regulatory requirements can be supported by machine-collected system evidence and which require complementary organizational, physical, or human assessment.

Appendix A

Language-Model Prompts

This appendix reports the prompt templates used by the language-model-based components of the experimental framework.

A.1 Security-Control Mapping Prompt

The following prompt was used to generate the model-generated security-control mappings used in the corresponding experimental configurations.

Return one JSON object only. Do not use markdown.

Task:

Map the directive control to the most relevant NIST SP
800–53 controls.

Directive control:

ID: {control_id}

Description: {control_description}

Candidate NIST controls:

{candidate_nist_controls}

Rules:

- Choose only IDs from the candidate list.
- Use 1 to 3 NIST controls.
- Prefer the closest semantic match.
- Do not invent control IDs.
- Do not copy the full candidate list.

Return exactly this JSON shape:

```
{  
  "nist_controls": [{"example_id"}],
```

```
    "rationale": "short reason"
}
```

A.2 Observation-Selection Prompt

The following prompt was used to generate the model-generated observation-selection rules used in the corresponding experimental configurations.

Return one JSON object only. Do not use markdown.

Task:

Select system observations that can provide evidence for
the mapped NIST objectives.

Directive control:

ID: {control_id}

Description: {control_description}

Mapped NIST controls:

{mapped_nist_controls}

NIST objectives:

{nist_objectives}

Available system observations:

{available_observations}

Rules:

- Select only observation IDs from the available system observations.
- Select only objective IDs from the NIST objectives list
- Do not invent new evidence fields.
- Do not provide evidence values.
- Use at most {max_observations} observations.

Return exactly this JSON shape:

```
{
  "observations": [
    {
      "id": "tls-configured",
      "objective_id": "SC-8::1"
    }
  ]
}
```

A.3 Compliance-Evaluation Prompt

The compliance evaluator used a system prompt and a user prompt. Unlike the previous two templates, these prompts were executed during the language-model-based evaluation runs.

A.3.1 System Prompt

You are a cybersecurity compliance analyst. Evaluate evidence against the control. Return one compact valid JSON object only.

A.3.2 User Prompt

Decide whether the evidence satisfies the cybersecurity control.

Control ID: {control_id}
Control description: {control_description}

NIST mapping:
{nist_mapping}

NIST assessment context:
{assessment_context}

Evidence:
{evidence}

Decision rules:

- PASS: evidence directly supports that the control is implemented.
- FAIL: evidence directly contradicts or shows the control is not implemented.
- UNKNOWN: evidence is missing, incomplete, ambiguous, or not relevant.

Important:

- NIST mapping and assessment context are reference material, not evidence.
- Use only the Evidence object as observed evidence.
- If Evidence is empty, return UNKNOWN.
- Do not list evidence fields that are not present in the Evidence object.

- Boolean true means the assessment objective is satisfied.
- Boolean false means the assessment objective is not satisfied and supports FAIL.
- Null means the evidence was unavailable or not collected.
- If any Evidence value is null and no Evidence value is false, return UNKNOWN.

Do not assume facts that are not present in the evidence.
 Explanation must cite the main evidence fields used.
 Keep explanation under 35 words.
 Keep remediation under 25 words.
 Use at most 4 used_evidence fields.
 For PASS, remediation should be an empty string.
 For FAIL or UNKNOWN, remediation must not be empty.
 Do not use markdown or code fences.

Return JSON:

```
{
  "decision": "",
  "used_evidence": [],
  "explanation": "",
  "remediation": ""
}
```

Bibliography

- [1] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach (4th Edition)*. Pearson, 2020.
- [2] A. M. Turing, “Computing Machinery and Intelligence,” *Mind*, vol. 59, pp. 433–460, 1950.
- [3] T. Mitchell, *Machine Learning*. McGraw Hill, 1997.
- [4] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.
- [5] R. Salakhutdinov, J. B. Tenenbaum, and A. Torralba, “Learning with Hierarchical-Deep Models,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1958–1971, 2013.
- [6] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, Y. Du, C. Yang, Y. Chen, Z. Chen, J. Jiang, R. Ren, Y. Li, X. Tang, Z. Liu, P. Liu, J.-Y. Nie, and J.-R. Wen, “A Survey of Large Language Models,” *Frontiers of Computer Science*, vol. 20, p. 2012627, 2026.
- [7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 6000–6010.
- [8] J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler, E. H. Chi, T. Hashimoto, O. Vinyals, P. Liang, J. Dean, and W. Fedus, “Emergent Abilities of Large Language Models,” *Transactions on Machine Learning Research*, 2022.
- [9] Y. Zhang, Y. Li, L. Cui, D. Cai, L. Liu, T. Fu, X. Huang, E. Zhao, Y. Zhang, Y. Chen, L. Wang, A. T. Luu, W. Bi, F. Shi, and S. Shi, “Siren’s Song in the AI Ocean: A Survey on Hallucination in Large Language Models,” *Computational Linguistics*, vol. 51, no. 4, pp. 1373–1418, 2025.

- [10] M. T. R. Laskar, S. Alqahtani, M. S. Bari, M. Rahman, M. A. M. Khan, H. Khan, I. Jahan, A. Bhuiyan, C. W. Tan, M. R. Parvez, E. Hoque, S. Joty, and J. X. Huang, “A Systematic Survey and Critical Review on Evaluating Large Language Models: Challenges, Limitations, and Recommendations,” in *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, 2024, pp. 13 785–13 816.
- [11] L. M. Kölbl, L. Poss, and S. Schöning, “Context is key for cybersecurity: leveraging external knowledge for process model explanation via LLMs,” *Int. J. Inf. Secur.*, vol. 25, no. 4, 2026.
- [12] H. Lee, S. J. Joo, C. Kim, J. Jang, D. Kim, K.-W. On, and M. Seo, “How Well Do Large Language Models Truly Ground?” In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, Association for Computational Linguistics, 2024, pp. 2437–2465.
- [13] N. Capuano, G. Fenza, V. Loia, and C. Stanzione, “Explainable Artificial Intelligence in CyberSecurity: A Survey,” *IEEE Access*, vol. 10, pp. 93 575–93 600, 2022.
- [14] P. Linardatos, V. Papastefanopoulos, and S. B. Kotsiantis, “Explainable AI: A Review of Machine Learning Interpretability Methods,” *Entropy*, vol. 23, 2021.
- [15] P. J. Phillips, C. Hahn, P. Fontana, D. Broniatowski, and M. Przybicki, “Four Principles of Explainable Artificial Intelligence,” National Institute of Standards and Technology, Tech. Rep., 2021.
- [16] G. Schwalbe and B. Finzel, “A comprehensive taxonomy for explainable artificial intelligence: A systematic survey of surveys on methods and concepts,” *Data Mining and Knowledge Discovery*, vol. 38, pp. 3043–3101, 2024.
- [17] M. T. Ribeiro, S. Singh, and C. Guestrin, “”Why Should I Trust You?”: Explaining the predictions of any classifier,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Association for Computing Machinery, 2016, pp. 1135–1144.
- [18] S. M. Lundberg and S.-I. Lee, “A Unified Approach to Interpreting Model Predictions,” in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 4768–4777.
- [19] M. T. Ribeiro, S. Singh, and C. Guestrin, “Anchors: high-precision model-agnostic explanations,” in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI Press, 2018.

- [20] R. Guidotti, A. Monreale, S. Ruggieri, D. Pedreschi, F. Turini, and F. Giannotti, “Local Rule-Based Explanations of Black Box Decision Systems,” *ArXiv*, vol. abs/1805.10820, 2018.
- [21] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, “Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization,” *Int. J. Comput. Vision*, vol. 128, no. 2, pp. 336–359, 2020.
- [22] A. Dhurandhar, P.-Y. Chen, R. Luss, C.-C. Tu, P. Ting, K. Shanmugam, and P. Das, “Explanations based on the Missing: Towards Contrastive Explanations with Pertinent Negatives,” in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, Curran Associates Inc., 2018, pp. 590–601.
- [23] C. Temel, “Towards Trustworthy Autonomous Robots: An Explainable AI-Based Decision Framework,” in *SoutheastCon 2026*, 2026, pp. 1–6.
- [24] M. A. Shareef, “Engineering Trust in AI Systems: A Data-Layer Framework for Explainability and Auditability,” *The USA Journals TAJIR*, vol. 8, pp. 83–89, 2026.
- [25] A. Goncalves and A. Correia, “Engineering Explainable AI Systems for GDPR-Aligned Decision Transparency: A Modular Framework for Continuous Compliance,” *Journal of Cybersecurity and Privacy*, vol. 6, no. 1, p. 7, 2026.
- [26] International Telecommunication Union, “Overview of Cybersecurity,” International Telecommunication Union, Tech. Rep. X.1205, 2008.
- [27] Translation Bureau, “Emergency Management Vocabulary,” Public Works and Government Services Canada, Tech. Rep. 281, 2012.
- [28] J. Jang-Jaccard and S. Nepal, “A survey of emerging threats in cybersecurity,” *Journal of Computer and System Sciences*, vol. 80, no. 5, pp. 973–993, 2014.
- [29] H. S. Lin, A. Z. Spector, P. G. Neumann, and S. E. Goodman, “Toward a Safer and More Secure Cyberspace,” *Commun. ACM*, vol. 50, no. 10, pp. 128–133, 2007.
- [30] C. Pascoe, S. Quinn, and K. Scarfone, “The NIST Cybersecurity Framework (CSF) 2.0,” National Institute of Standards and Technology, Tech. Rep., 2024.
- [31] A. Folorunso, I. Wada, B. Samuel, and V. Mohammed, “Security compliance and its implication for cybersecurity,” *World Journal of Advanced Research and Reviews*, vol. 24, pp. 2105–2121, Oct. 2024.
- [32] R. Stevens, J. Dykstra, W. Everette, J. Chapman, G. Bladow, A. Farmer, K. Halliday, and M. Mazurek, “Compliance Cautions: Investigating Security Issues Associated with U.S. Digital-Security Standards,” in *Proceedings 2020 Network and Distributed System Security Symposium*, 2020, pp. 1–18.

- [33] W. Wang, S. M. Sadjadi, and N. Rishe, “A survey of major cybersecurity compliance frameworks,” in *Proceedings of the IEEE 10th Conference on Big Data Security on Cloud (BigDataSecurity)*, 2024, pp. 23–34.
- [34] R. Foorthuis and R. Bos, “A framework for organizational compliance management tactics,” in *Advanced Information Systems Engineering Workshops*, Springer, 2011, pp. 259–268.
- [35] S. W. A. Hamdani, H. Abbas, A. Janjua, W. Shahid, M. Amjad, J. Malik, M. Murtaza, M. Atiquzzaman, and A. Khan, “Cybersecurity standards in the context of operating system: Practical aspects, analysis, and comparisons,” *ACM Computing Surveys*, vol. 54, pp. 1–36, 2021.
- [36] M. Mubarkoot, J. Altmann, M. Barzoki, B. Egger, and H. Lee, “Software Compliance Requirements, Factors, and Policies: A Systematic Literature Review,” *Computers & Security*, vol. 124, p. 102985, 2022.
- [37] M. Usman, M. Felderer, M. Unterkalmsteiner, E. Klotins, D. Mendez, and E. Alégroth, “Compliance requirements in large-scale software development: An industrial case study,” in *Proceedings of the International Conference on Product-Focused Software Process Improvement*, Springer, 2020, pp. 385–401.
- [38] M. Sadok, S. L. Alter, and P. M. Bednar, “It is not my job: Exploring the disconnect between corporate security policies and actual security practices in smes,” *Information and Computer Security*, vol. 28, pp. 467–483, 2020.
- [39] NIST Joint Task Force, “Security and Privacy Controls for Information Systems and Organizations,” National Institute of Standards and Technology, NIST Special Publication (SP) 800-53 Revision 5, 2020.
- [40] United States Office of Management and Budget, *Circular no. a-130: Managing information as a strategic resource*, Appendix I, 2016.
- [41] NATO Security Committee, “Directive on the Security of Information,” North Atlantic Treaty Organization, Tech. Rep. AC/35-D/2002-REV4, 2012.
- [42] W. Piez, “The Open Security Controls Assessment Language (OSCAL): Schema and Metaschema,” in *Proceedings of Balisage: The Markup Conference 2019*, vol. 23, 2019.
- [43] M. Adawadkar, “The Evolution of Generative and Agentic AI: From Rule-Based Systems to Autonomous Intelligence,” *American Journal of Technology Advancement*, vol. 2, no. 11, pp. 1–11, 2025.
- [44] V. Selvaraj, “Cognitive AI for Proactive Cyber Defense: A Rule-Based Multi-Agent Approach,” in *Proceedings of the 2025 IEEE International Conference on Recent Advances in Computing and Systems (REACS)*, 2025, pp. 1–9.

- [45] S. Srinivas, B. Kirk, J. Zendejas, M. Espino, M. Boskovich, A. Bari, K. Dajani, and N. Alzahrani, “AI-Augmented SOC: A Survey of LLMs and Agents for Security Automation,” *Journal of Cybersecurity and Privacy*, vol. 5, p. 95, 2025.
- [46] J. Zhang, H. Bu, H. Wen, Y. Liu, H. Fei, R. Xi, L. Li, Y. Yang, H. Zhu, and D. Meng, “When LLMs Meet Cybersecurity: A Systematic Literature Review,” *Cybersecurity*, vol. 8, p. 55, 2025.
- [47] Y. Meng, L. Tang, F. Yu, J. Jia, G. Yan, P. Yang, and Z. Xi, “Uncovering Vulnerabilities of LLM-Assisted Cyber Threat Intelligence,” *arXiv e-prints*, 2025. arXiv: 2509.23573.
- [48] S. Banerjee, A. Agarwal, and S. Singla, “LLMs Will Always Hallucinate, and We Need to Live with This,” in *Intelligent Systems and Applications*, Springer Nature, 2025, pp. 624–648.
- [49] Q. Lyu, M. Apidianaki, and C. Callison-Burch, “Towards Faithful Model Explanation in NLP: A Survey,” *Computational Linguistics*, vol. 50, no. 2, pp. 657–723, 2024.
- [50] S. Chaduvula, J. Ho, K. Kim, A. Narayanan, A. Y. Radwan, M. Alinoori, M. Garg, D. Ramachandram, and S. Raza, “From Features to Actions: Explainability in Traditional and Agentic AI Systems,” *arXiv e-prints*, 2026. arXiv: 2602.06841.
- [51] T. Yamaguchi, Y. Zhou, M. Ryo, and K. Katsura, “Agentic Explainable Artificial Intelligence (Agentic XAI) Approach To Explore Better Explanation,” *arXiv e-prints*, 2026. arXiv: 2512.21066.
- [52] H. Rashkin, V. Nikolaev, M. Lamm, L. Aroyo, M. Collins, D. Das, S. Petrov, G. S. Tomar, I. Turc, and D. Reitter, “Measuring Attribution in Natural Language Generation Models,” *Computational Linguistics*, vol. 49, no. 4, pp. 777–840, 2023.
- [53] K. Dempsey, P. Eavy, and G. Moore, “Automation Support for Security Control Assessments: Volume 1: Overview,” National Institute of Standards and Technology, NIST Interagency Report 8011, Volume 1, 2017.
- [54] L. O. Nweke and P. K. Yeng, “Compliance-as-Code for AI-Driven Identity Systems: Clause-to-Control Traceability and Machine-Readable Evidence,” *IEEE Access*, vol. 14, pp. 28 258–28 281, 2026.
- [55] L. Nogueira and A. Resende, “Reusing Policy-as-Code Across CI/CD and Kubernetes Admission Control: An Empirical Assessment of Governance Consistency,” *Computers*, vol. 15, no. 7, 2026.
- [56] Open Policy Agent, *Open policy agent documentation*, <https://www.openpolicyagent.org/docs>, Accessed: 2026-09-15, 2026.
- [57] A. Verdet, M. Hamdaqa, L. D. Silva, and F. Khomh, “Assessing the adoption of security policies by developers in terraform across different cloud providers,” *Empirical Softw. Engg.*, vol. 30, no. 3, 2025.

- [58] Prisma Cloud, *Checkov*, <https://github.com/bridgecrewio/checkov>, Accessed: 2026-09-14, 2026.
- [59] Aqua Security, *Tfsec*, <https://github.com/aquasecurity/tfsec>, Accessed: 2026-09-14, 2025.
- [60] A. Mohammed, “AI in Cybersecurity: Enhancing Audits and Compliance Automation,” *Innovative Computer Sciences Journal*, vol. 7, p. 10, 2021.
- [61] S. Hassani, M. Sabetzadeh, D. Amyot, and J. Liao, “Rethinking Legal Compliance Automation: Opportunities with Large Language Models,” in *Proceedings of the 2024 IEEE 32nd International Requirements Engineering Conference (RE)*, 2024, pp. 432–440.
- [62] O. Amaral, S. Abualhaija, D. Torre, M. Sabetzadeh, and L. C. Briand, “AI-Enabled Automation for Completeness Checking of Privacy Policies,” *IEEE Transactions on Software Engineering*, vol. 48, pp. 4647–4674, 2022.
- [63] O. A. Cejas, M. I. Azeem, S. Abualhaija, and L. C. Briand, “NLP-Based Automated Compliance Checking of Data Processing Agreements Against GDPR,” *IEEE Transactions on Software Engineering*, vol. 49, pp. 4282–4303, 2023.
- [64] L. Alevizos, “Automated cybersecurity compliance and threat response using AI, blockchain and smart contracts,” *International Journal of Information Technology*, vol. 17, pp. 767–781, 2024.
- [65] M. Ameer, B. Brik, and A. Ksentini, “EUROCOMPLY: Enabling Zero-Touch AI Compliance Auditing via LLM-Based Agentic AI,” *IEEE Communications Magazine*, vol. 64, pp. 82–88, 2026.
- [66] Pallets Projects, *Flask documentation*, <https://flask.palletsprojects.com/>, Accessed: 2026-09-07, 2026.
- [67] PostgreSQL Global Development Group, *Postgresql documentation*, <https://www.postgresql.org/docs/>, Accessed: 2026-09-07, 2026.
- [68] F5, Inc., *Nginx reverse proxy documentation*, <https://docs.nginx.com/nginx/admin-guide/web-server/reverse-proxy/>, Accessed: 2026-09-07, 2026.
- [69] Docker Inc., *Docker documentation: What is docker?* <https://docs.docker.com/get-started/docker-overview/>, Accessed: 2026-09-07, 2026.
- [70] HashiCorp, *Terraform documentation*, <https://developer.hashicorp.com/terraform/docs>, Accessed: 2026-09-07, 2026.
- [71] NATO Consultation, Command and Control Board (C3B), “Technical and Implementation Directive on CIS Security,” North Atlantic Treaty Organization, Tech. Rep. AC/322-D/0048-REV3 (INV), 2019.

- [72] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, and A. Rush, “Transformers: State-of-the-Art Natural Language Processing,” in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Association for Computational Linguistics, 2020, pp. 38–45.