

**SANTA CATARINA STATE UNIVERSITY - UDESC
COLLEGE OF TECHNOLOGICAL SCIENCE - CCT
GRADUATE PROGRAM IN APPLIED COMPUTING - PPGCAP**

MILTON PEDRO PAGLIUSO NETO

**SAFEPROBE: A PYTHON TOOLKIT FOR PROBING THE SAFETY
ALIGNMENT OF LARGE LANGUAGE MODELS**

JOINVILLE

2026

MILTON PEDRO PAGLIUSO NETO

**SAFEPROBE: A PYTHON TOOLKIT FOR PROBING THE SAFETY
ALIGNMENT OF LARGE LANGUAGE MODELS**

Master thesis presented to the Graduate Program in Applied Computing of the College of Technological Science from the Santa Catarina State University, as a partial requisite for receiving the Master's degree in Applied Computing.

Supervisor: Dr. Charles Christian Miers

JOINVILLE

2026

Neto, Milton Pedro Pagliuso

SafeProbe: A Python Toolkit for Probing the Safety Alignment of Large Language Models/ Milton Pedro Pagliuso Neto. – 2026.
89 p.

Orientador: Dr. Charles Christian Miers

Dissertação (mestrado) – Universidade do Estado de Santa Catarina, Centro de Ciências Tecnológicas, Programa de Pós-Graduação em Computação Aplicada, Joinville, 2026.

1. AI. 2. LLMs. 3. CyberSecurity. 4. AML. I. Miers, Charles Christian. II. Universidade do Estado de Santa Catarina, Centro de Ciências Tecnológicas, Programa de Pós-Graduação em Computação Aplicada. III. SafeProbe: A Python Toolkit for Probing the Safety Alignment of Large Language Models.

Milton Pedro Pagliuso Neto

**SafeProbe: A Python Toolkit for Probing the Safety Alignment of Large
Language Models**

Master thesis presented to the Graduate Program in Applied Computing of the College of Technological Science from the Santa Catarina State University, as a partial requisite for receiving the **Master's degree in Applied Computing**.

Master Thesis Committee:

Prof. Dr. Charles Christian Miers
Santa Catarina State University
President/Advisor

Prof. Dr. Guilherme Piêgas Koslovski
Santa Catarina State University
Board member

Prof. Dr. Victor Takashi Hayashi
University of São Paulo
Board member

Joinville, 29th June 2026

To my loving wife, family, friends, advisor
and mentor, this work is dedicated to your
unwavering support.

ACKNOWLEDGMENTS

This thesis is the aftermath of many days of work, overcoming hardships and aiming at always having enough strength, mindfulness and love into each dream that hard work can achieve.

I would like to thank my advisor, Professor Charles C. Miers, for all the support, caring and belief throughout this step in my life, pursuing a master's degree, his direction has allowed me to go further than I could foresee in my academic, professional and personal life.

I would also like to praise all of the amazing professors from the University of São Paulo which helped me find so many opportunities to develop myself on numerous projects.

To my wife Anna Julia, who has been the biggest supporter since the beginning of this journey, believing in my success and helping me in every step of the way, being my daily reminder that I had all the tools to achieve anything I'd dream about.

To all my friends, who have been my support network through the hard times, good times and everything that came my way. You remind me that although work is an important part of life, we should spare a special moment for good laughter and to care for ourselves.

I'm grateful to God for all the strength and countless blessings He has brought into my life during these challenging years, may this thesis be a part of everything I'm here to achieve in this life.

At last, to my projects colleagues that were a key part of my academic growth, allowing my mindset to bloom through different points-of-view and being a part of many of my research efforts.

We cannot be satisfied with merely calling for the moralization of machines — the so-called “alignment” of AI with human values — without also having the courage to insist on a further condition: the possibility of openly discussing the ethical frameworks involved and subjecting them to shared standards of social justice. Otherwise, those who control AI will impose their own moral vision, which will become the invisible infrastructure of these systems (Pope Leo XIV, 2026, no. 107).

ABSTRACT

Large Language Models (LLMs) are deployed as automated services in customer support, content generation, and decision-making, which exposes their safety alignment to adversaries that interact through query access alone. A recurring methodological limitation in the literature is that the Attack Success Rate (ASR) used to quantify this alignment is typically derived from a single classifier or LLM judge, with no measurement of how stable that verdict would be under a different judge, which undermines cross-paper and cross-model comparability. This work presents SafeProbe, an open-source, pip-installable Python toolkit that evaluates the safety alignment of LLMs at inference time under a query-access threat model through a four-stage pipeline of attack, consolidation, multi-judge classification, and reporting. The toolkit integrates a unified adapter layer that places proprietary and open-weight models under a single protocol, four attack modules of increasing sophistication (PromptMap, CipherChat, PAIR, and a novel Composite technique that exhaustively pairs Competing Objectives with Mismatched Generalization strategies), and a multi-judge evaluation layer that runs three architecturally distinct judges over each response and reports inter-rater agreement using Cohen’s and Fleiss’ kappa. SafeProbe was applied to five target models across three jailbreak benchmarks, producing 7,800 (prompt, response) pairs, and the automated judges were validated against a human-annotated reference of 200 items. The results show that no single attack technique is universally most effective, that reported ASR values are judge-conditioned rather than model-intrinsic (varying by up to a factor of 14 on the same target), and that the per-pair Composite maxima exceed the aggregate by a factor of three to six. These findings support reporting an inter-rater agreement statistic alongside every published jailbreak ASR, in the same way that classification accuracy is conventionally accompanied by a confidence interval.

Keywords: large language models, safety alignment, red-teaming, adversarial machine learning, LLM-as-a-judge.

RESUMO

Os Grandes Modelos de Linguagem (LLMs) são implantados como serviços automatizados em atendimento ao cliente, geração de conteúdo e apoio à decisão, o que expõe seu alinhamento de segurança a adversários que interagem apenas por meio de acesso via consulta (*query access*). Uma limitação metodológica recorrente na literatura é que a Taxa de Sucesso de Ataque (ASR), usada para quantificar esse alinhamento, costuma ser obtida a partir de um único classificador ou modelo juiz, sem qualquer medição de quão estável seria esse veredito sob um juiz diferente, o que compromete a comparabilidade entre trabalhos e entre modelos. Esta dissertação apresenta o SafeProbe, um *toolkit* Python de código aberto e instalável via *pip* que avalia o alinhamento de segurança de LLMs em tempo de inferência sob um modelo de ameaça de acesso via consulta, por meio de um *pipeline* de quatro estágios: ataque, consolidação, classificação multi-juiz e geração de relatórios. O *toolkit* integra uma camada de adaptadores unificada que coloca modelos proprietários e de pesos abertos sob um único protocolo, quatro módulos de ataque de sofisticação crescente (PromptMap, CipherChat, PAIR e uma técnica *Composite* inédita que combina exaustivamente estratégias de Objetivos Concorrentes com Generalização Incompatível) e uma camada de avaliação multi-juiz que executa três juízes arquiteturalmente distintos sobre cada resposta e apresenta a concordância entre avaliadores por meio dos coeficientes *kappa* de Cohen e de Fleiss. O SafeProbe foi aplicado a cinco modelos-alvo em três *benchmarks* de *jailbreak*, gerando 7.800 pares (*prompt*, resposta), e os juízes automatizados foram validados frente a uma referência anotada por humanos com 200 itens. Os resultados mostram que nenhuma técnica de ataque é universalmente mais eficaz, que os valores de ASR relatados na literatura dependem do juiz e não são intrínsecos ao modelo (variando por um fator de até 14 no mesmo alvo) e que os máximos por par do ataque *Composite* superam o valor agregado por um fator de três a seis. Esses achados sustentam a recomendação de que toda ASR de *jailbreak* publicada seja acompanhada de uma estatística de concordância entre avaliadores, da mesma forma que a acurácia de classificação é convencionalmente acompanhada de um intervalo de confiança.

Palavras-chave: grandes modelos de linguagem, alinhamento de segurança, *red-teaming*, aprendizado de máquina adversário, LLM como juiz.

LIST OF FIGURES

Figure 1 – Taxonomy of attacks on GenArtificial Intelligence (AI) systems. . . .	29
Figure 2 – Large Language Model (LLM) enterprise adoption reference architecture.	36
Figure 3 – SafeProbe architecture for LLM safety alignment evaluation.	46
Figure 4 – Sequence diagram for the attack effectiveness and judge validity experiment.	52
Figure 5 – Testbed Setup.	54
Figure 6 – Attack Success Rate by Target Model and Attack Technique.	67
Figure 7 – Composite (CO X MG) Combinations per Target Model.	69
Figure 8 – Overall Attack Success Rate (%) per Target Model Across Judges. .	71
Figure 9 – Inter-rater Agreement (Cohen’s k and Fleiss’ k) Between Judges per Target Model.	72
Figure 10 – Per-judge Agreement with Human Ground Truth Across Evaluation Metrics.	75

LIST OF TABLES

Table 1 – Comparison of machine learning paradigms	19
Table 2 – Inclusion and Exclusion Criteria.	40
Table 3 – Overview of LLM safety evaluation frameworks identified in the literature.	41
Table 4 – Selected frameworks versus functional requirements.	44
Table 5 – Target models evaluated.	50
Table 6 – Attack Success Rate by target model and attack technique. Cells are formatted as “ M / HB”, where M is the majority-of-3 verdict and HB is HarmBench’s individual verdict (the judge most aligned with humans, see Section 4.2.3). Attack Success Rate (ASR) is computed only over decisive verdicts.	66
Table 7 – Top-three Composite ($CO \times MG$) combinations per target, ranked by majority-of-3 ASR. Each cell reports the combination (top line), its majority ASR (middle line), and HarmBench’s individual ASR on the same pair (bottom line).	68
Table 8 – Per-judge overall ASR (%) by target. ASR is computed only over decisive verdicts (YES/NO). The Majority column reports the consensus verdict (at least 2 of 3 judges agree on YES); rows for which judges split 1-1-1 or for which two or more judges return AMBIGUOUS are excluded from the majority denominator.	71
Table 9 – Inter-rater agreement per target: pairwise Cohen’s κ between the three judges and overall Fleiss’ κ . All κ values are computed over rows for which all three judges produced a decisive verdict, so the three pairwise values and Fleiss’ κ share a common denominator. Interpretation labels follow the Landis and Koch scale.	72
Table 10 – Per-judge agreement with the human-majority ground truth on the 200-item validation sample. P: precision; R: recall; F1: harmonic mean; Acc: accuracy; κ : Cohen’s kappa with the human-majority label. The Majority(2/3) row reports the consensus judge verdict against the human-majority. All metrics are computed over rows for which both the judge (or judge-majority) and the human-majority verdicts are decisive. . . .	74

LIST OF ABBREVIATIONS AND ACRONYMS

AI	Artificial Intelligence
RAG	Retrieval-Augmented Generation
RLHF	Reinforcement Learning from Human Feedback
RNN	Recurrent Neural Network
RQ	Research Question
AML	Adversarial Machine Learning
ANN	Artificial Neural Networks
ASR	Attack Success Rate
API	Application Programming Interface
CoT	Chain-of-Thought
CLI	Command-Line Interface
DNN	Deep Neural Networks
FR	Functional Requirements
LLM	Large Language Model
LabP2D	Laboratory of Parallel and Distributed Processing
MoE	Mixture of Experts
NFR	Non-Functional Requirements
NLP	Natural Language Processing
PPGCAP	Graduate Program in Applied Computing
SSH	Secure Shell
TCP	Transmission Control Protocol
TPS	Transactions Per Second
UDESC	Universidade do Estado de Santa Catarina
UDP	User Datagram Protocol
VM	Virtual Machine
ANN	Artificial Neural Network
DNN	Deep Neural Network
NIST	National Institute of Standards and Technology
CIA	Confidentiality, Integrity, and Availability
GAN	Generative Adversarial Network
VAE	Variational Autoencoder

CONTENTS

1	INTRODUCTION	14
2	FUNDAMENTAL CONCEPTS	17
2.1	Artificial Intelligence	18
2.2	AI and Cybersecurity	23
2.3	Adversarial AI	25
2.4	NLP and LLMs	29
2.5	Problem and opportunities	31
2.6	Chapter considerations	33
3	REQUIREMENTS AND PROPOSAL	35
3.1	Scenario	35
3.2	Prerequisites	37
3.3	Requirements	37
3.4	Related Works	39
3.4.1	Related Work Selection	39
3.4.2	Inclusion and Exclusion Criteria	40
3.4.3	Search Results and Analysis	40
3.4.4	Framework Selection Justification	42
3.4.5	Comparative Analysis	44
3.5	Proposal	46
3.6	Test Plan	49
3.6.1	Experimental Design	50
3.6.2	Experiment: Attack Effectiveness and Judge Validity	51
3.7	Testbed Setup	53
3.8	Chapter Considerations	54
4	IMPLEMENTATION AND RESULTS	56
4.1	Implementation	56
4.1.1	Software Architecture	56
4.1.2	Model Adapters (FR1)	57
4.1.3	Attack Modules (FR2)	58
4.1.3.1	PromptMap: Rule-Based Adversarial Mapping	58
4.1.3.2	PAIR: Prompt Automatic Iterative Refinement	59
4.1.3.3	CipherChat: Encoding-Based Attacks	60

4.1.3.4	Composite: Exhaustive CO × MG Pairing	60
4.1.4	Multi-Judge Evaluation Layer (FR3 and FR4)	61
4.1.4.1	Judge Backends	61
4.1.4.2	Multi-Judge Orchestration	63
4.1.4.3	Inter-Rater Agreement	63
4.1.5	Metrics Computation	64
4.1.6	Report Generation	64
4.1.7	Command-Line Interface	65
4.2	Results	65
4.2.1	Attack Effectiveness	66
4.2.2	Cross-Judge Agreement	70
4.2.3	Human Validation	73
4.2.4	Discussion	77
5	CONSIDERATIONS & FUTURE WORK	79
5.1	Publications	80
5.2	Future Work	82
5.2.1	AI Usage Declaration	83
	BIBLIOGRAPHY	84

1 INTRODUCTION

The AI field has undergone accelerated development through deep learning models built on distinct architectures and machine learning paradigms (LECUN; BENGIO; HINTON, 2015). These models attain results comparable to those of human experts in tasks such as image classification, object detection, and text generation. As a consequence, application domains such as customer service and sales have incorporated AI tools into their systems, with the aim of optimizing workforce allocation and operational costs.

The integration of these models into systems that interact directly with users has been accompanied by a growing number of incidents involving such systems and the people who use them. These incidents range from the generation of factually incorrect information and the reproduction of harmful stereotypes or biased behavior to training-data leakage that compromises user privacy (ZHANG et al., 2024b). Such occurrences indicate that a relevant portion of these integrations has been conducted without sufficient attention to the safety aspects inherent in deploying the models. The vulnerabilities of deep learning models can emerge at distinct stages of the model lifecycle, including training, fine-tuning, and inference, and require systematic analysis to establish the minimum safety standards necessary for secure deployment environments.

Organizations that adopt deep learning models, such as LLMs, as automated tools for customer service, content generation, decision-making, data analysis, and recommendation frequently employ pre-trained proprietary models without evaluating their built-in safeguards (SMUHA, 2021). This conduct is associated with the appeal of rapid deployment and with the assumed sophistication of these models, which leads to the omission of safety evaluations and risk assessments. The incidents reported in deployed systems have motivated governments to develop laws and regulations for the safe use of AI systems and to establish accountability frameworks for their deployment (JOBIN; IENCA; VAYENA, 2019). Within this context, the present work addresses the evaluation of the safety standards of LLMs, given their adoption across the domains listed above, through an assessment of the safety alignment of these models.

This work examines the safety alignment of LLMs by focusing on the vulnerabilities present at the inference stage, under a threat model in which an adversary interacts with the model exclusively through query access. The examination covers the reproduction of adversarial attack techniques against LLMs, the systematic classification of the resulting responses, and the measurement of model robustness through quan-

titative metrics. To support this examination, an evaluation toolkit named SafeProbe is proposed, integrating a set of red-teaming techniques with an evaluation pipeline that classifies model responses using multiple independent judges and reports the agreement among them.

The motivation for this work derives from a methodological limitation identified in existing safety evaluation approaches: the ASR values reported in the literature are obtained from a single classifier or judge model, with no measurement of how stable those values would be under a different judge. This limitation compromises both the comparison of results across studies and the assessment of the safety posture of distinct models. The treatment of the judge choice as a measurable variable, rather than as a fixed component, constitutes the central concern that guides the development of the proposed toolkit.

The central concern stated above is investigated through four research questions. Research Question (RQ)1 asks whether, under a query-access threat model, a single adversarial attack technique dominates across target models, or whether the most effective technique is target-dependent. RQ2 asks whether jailbreak ASR values are model-intrinsic or conditioned on the choice of automated judge, and, when they are judge-conditioned, how large the effect is. RQ3 asks whether exhaustively pairing competing-objective with mismatched-generalization strategies exposes target-specific vulnerabilities that single-technique or aggregate reporting conceals. RQ4 asks which of a set of architecturally distinct judges most closely approximates human judgments of jailbreak success, and how that alignment reframes the per-judge ASR values.

These questions correspond to four hypotheses that Chapter 4 evaluates against empirical measurements. The first hypothesis states that no single attack technique is universally most effective, the strongest technique varying by target. The second hypothesis states that ASR is judge-conditioned, so that the same set of (prompt, response) pairs yields ASR values that differ across judges by up to a factor of fourteen on a single target. The third hypothesis states that the strongest per-pair combination of the composite attack exceeds the aggregate composite ASR by a factor of three to six. The fourth hypothesis states that architecturally distinct judges differ in their alignment with human judgments, so that the judge reporting the highest ASR is not the judge most aligned with human raters.

The general objective of this work is to develop an evaluation toolkit capable of assessing the safety alignment of LLMs subjected to diverse attack vectors under a query-access threat model, establishing a reproducible methodology for testing models before their deployment as integrated services. To attain this general objective, the following specific objectives are defined:

- to survey existing LLM safety evaluation frameworks and identify the requirements that an inference-time evaluation toolkit must satisfy;
- to implement a unified adapter layer that allows the evaluation of proprietary and open-weight models under a single protocol;
- to implement a set of attack modules covering distinct levels of sophistication, including a composite technique that pairs competing-objective and mismatched-generalization strategies;
- to implement a multi-judge evaluation layer that classifies model responses through semantic intent analysis and reports the inter-rater agreement; and
- to validate the automated judges against a human-annotated reference, quantifying the alignment between each judge and the human evaluators.

The execution of this work combines two methodological approaches. The first consists of a systematic mapping of the literature, conducted according to established guidelines for systematic reviews, which supports the identification of related frameworks and the derivation of the functional and non-functional requirements. The second consists of empirical experimentation, in which the proposed toolkit is implemented and applied to a set of target models within a controlled testbed, and the obtained measurements are analyzed against an external human-annotated reference. This combination grounds both the requirements that guide the design of the toolkit and the conclusions drawn from its application. The Functional Requirements (FR) and Non-Functional Requirements (NFR) derived in Chapter 3 specify the capabilities the toolkit must satisfy in order to answer the four research questions stated in this chapter; these requirements are the engineering counterpart of the research questions, and not a substitute for them.

The remainder of this work is organized so that the theoretical foundation precedes the definition of the problem, the definition of the problem precedes the proposed solution, and the proposed solution precedes its empirical assessment, allowing the reader to follow the reasoning that connects the identified gap to the obtained results. Chapter 2 presents the fundamental concepts of AI, adversarial AI, and LLMs, and establishes the research problem together with its motivations. Chapter 3 derives the FR and NFR, positions the work against related frameworks through the systematic mapping, details the proposed toolkit, and specifies both the test plan and the testbed. Chapter 4 describes the implementation of the toolkit and reports the experimental results, covering attack effectiveness, cross-judge agreement, and the human-validation study. Chapter 5 presents the final considerations and the directions identified for future work.

2 FUNDAMENTAL CONCEPTS

The field of AI has been in the spotlight for a number of reasons, with new technologies changing the way people handle day-to-day tasks. The growth of AI has been accompanied by computational resources that enable these systems to perform complex tasks without explicit programming. This technological advancement presents opportunities, but it also introduces significant security challenges that must be addressed. There is a clear necessity for robust security frameworks that can protect AI systems while maximizing their benefits. The availability of AI models for the wide public has opened new horizons for various fields and applications. Instead of relying on rule-based systems, new technologies involve complex approaches such as machine learning and deep learning to solve increasingly difficult problems. This enables improved decision-making, pattern recognition, and automation, making AI useful for applications across numerous domains.

In order to understand the problems presented in this work, it is necessary to examine relevant aspects related to AI systems and their security implications. This chapter presents the fundamentals of AI, covering its history, approaches, and recent developments in the field (Section 2.1). Then the cybersecurity landscape is explored, highlighting the principles, practices, and technologies designed to protect digital systems from unauthorized access and attacks (Section 2.2). With the growing deployment of AI systems, the field of adversarial AI emerges to understand, develop, and mitigate techniques that can compromise or defend AI systems (Section 2.3). Finally, natural language processing and large language models are examined as a specific domain where security challenges are particularly significant (Section 2.4).

Based on this foundation, the growth of AI applications and their integration into critical systems present research opportunities to ensure robust security measures for these technologies. Security frameworks must address the unique vulnerabilities of AI systems. With the development of defense mechanisms, better security postures can be achieved while maintaining the usability of AI systems (Section 2.5).

In the context addressed, researchers and practitioners have begun to explore the security challenges of large language models and other AI systems. There remains research potential to develop standardized evaluation methodologies and metrics that can measure the robustness of these systems, providing guidance for developers deploying AI in different contexts.

2.1 ARTIFICIAL INTELLIGENCE

AI models have changed the way most businesses handle their processes and manage their problems, leveraging their decision-making ability. A large set of fields in the industry have been directly impacted by different tools that could optimize their results, create documents, images and presentations powered by AI technologies. Hence, AI has grown substantially in the last decade, along with a set of new powerful processing resources.

The field of AI is a computer science sub-field that focuses on the development of autonomous systems that are capable of solving different problems without being explicitly programmed for it. The history of AI can be traced back to the 1950s when Alan Turing proposed the famous Turing Test, which evaluates a machine's ability to exhibit intelligent behavior equivalent to, or indistinguishable from, that of a human (TURING, 1950). The field has experienced periods of rapid growth interspersed with "AI winters" when progress and funding diminished due to unmet expectations (RUSSELL; NORVIG, 2010). The resurgence of AI can be attributed to three key factors: the availability of vast amounts of data, significant improvements in computational power, and advancements in algorithm design, particularly in neural networks and deep learning approaches (LECUN; BENGIO; HINTON, 2015). The landscape of AI is far from monolithic, it represents a set of interconnected approaches and methodologies that have evolved over decades of research and practical application.

There is a wide spectrum of complexity on the set of approaches found in different systems. There were rule-based expert systems that followed predefined logical structures to solve domain-specific problems, relied on human-encoded knowledge and lacked the ability to learn from new data. In the middle of this spectrum are machine learning approaches that can identify patterns and make predictions, but require human intervention in feature selection and model architecture. Modern approaches involve deep learning systems capable of end-to-end learning with minimal human intervention, extracting features and representations directly from raw data. The technological evolution on how intelligence is approached changed the way different solutions are built and shows a paradigm shift in these computational systems.

The field of machine learning, a fundamental approach on modern AI systems, enables mathematical models to learn from data rather than following explicitly programmed instructions. Within machine learning, supervised learning, unsupervised learning, and reinforcement learning paradigms are found (MURPHY, 2012). In the context of AI, learning represents a statistical process of parameter optimization, guided by objective functions that measure performance against desired outcomes.

The design behind neural networks has evolved into sophisticated architec-

Table 1 – Comparison of machine learning paradigms

Paradigm	Training Data	Learning Approach	Applications
Supervised Learning	Labeled input-output pairs (x_i, y_i)	Learn mapping function $f : X \rightarrow Y$ to predict labels for new inputs	Classification, regression, object detection
Unsupervised Learning	Unlabeled data $\{x_i\}$	Discover hidden patterns, structures, or representations in data	Clustering, dimensionality reduction, anomaly detection
Reinforcement Learning	Environment states, actions, and rewards	Learn optimal policy π^* that maximizes cumulative reward	Robotics, autonomous systems

tures capable of processing complex multimodal inputs. The architecture of these networks bears a conceptual, if not strictly biological, resemblance to neural structures in the human brain. The layers of artificial neurons can extract features from input data, transforming raw information into meaningful representations that support reasoning and prediction. This hierarchical feature extraction parallels theories of how the visual cortex processes information. Modern systems can identify patterns humans might miss but struggle with forms of common-sense reasoning that humans perform effortlessly.

In recent years, generative AI has emerged as a transformative approach within the broader AI field, distinctly different from traditional predictive AI systems (GOODFELLOW, 2020). The distinction lies in the different relationships between the models' output design and the data it processes. Predictive models analyze inputs to produce outputs that classify, categorize, or forecast, projecting from the known to the unknown along dimensions. Generative models synthesize new artifacts that exist somewhere in the possibility space defined by their training data. This ability to generate novel content enables the creation of synthetic environments for training other AI systems, augmenting limited datasets, and exploring scenarios that would be impractical or impossible to observe in reality.

LLMs are Natural Language Processing (NLP) foundational models overcoming traditional boundaries between natural language understanding and generation. These models are trained on a large corpora of data and have developed advanced internal representations of language that capture not just syntax and semantics, but aspects of pragmatics and discourse structure. The ability of these models to perform tasks for which they were not explicitly trained challenges conventional understandings of what constitutes learning and adaptation in computational systems.

The governance of AI presents challenges that extend beyond technical con-

siderations into legal, philosophical, and societal domains. As AI systems show great capability of problem-solving, their appropriate use becomes increasingly complex. The pace of AI development often outstrips the development of regulatory frameworks and ethical guidelines, creating scenarios where technologies enter widespread use before their implications are fully looked after.

One of the earliest approaches in generative AI were Markov chains, which operate on the principle that future states depend only on the current state, not on the sequence of events that preceded it (MARKOV, 1971). These mathematical models represent a foundational probabilistic model for sequential data generation, where states represent words or characters in tasks, with transitions indicating the probability of one word following another. It established important foundations for probabilistic modeling in AI, demonstrating how patterns could be extracted from data and used to generate new content that maintains some characteristics of the training corpus (MANNING; SCHÜTZE, 1999). The limited ability of Markov chains in capturing the semantic understanding motivated the development of new architectures.

The Artificial Neural Networks (ANNs) emerged as a biologically inspired computational paradigm presenting advanced generative capabilities (MCCULLOCH; PITTS, 1943). These networks consist of interconnected nodes in layers that process information through non-linear transformations, allowing them to learn patterns from data through iterative adjustment of parameters. Early implementations of ANNs for generative purposes included autoencoders, which learn to reconstruct input data by creating internal representations that capture the distribution of the training dataset (HINTON; SALAKHUTDINOV, 2006). The introduction of backpropagation algorithms enhanced the training process of neural networks, allowing more efficient learning of model parameters and enabling more complex network architecture (RUMELHART; HINTON; WILLIAMS, 1986).

Deep Neural Networks (DNNs) expanded these capabilities by introducing multiple processing layers, enabling the extraction of progressively more abstract features from input data (BENGIO, 2009). This depth allows networks to learn hierarchical representations, capturing further complex patterns. Several DNNs architectures have advanced generative AI capabilities:

1. Generative Adversarial Networks (GANs) introduced a novel training approach involving two competing networks: a generator that creates samples and a discriminator that evaluates them (GOODFELLOW et al., 2014). This adversarial process results in generators capable of producing highly realistic outputs, particularly in domains such as image generation.

2. Variational Autoencoders (VAEs) combine neural networks with principles from Bayesian inference (KINGMA; WELLING, 2013). These models learn a probabilistic mapping between data and a latent space, enabling both generation of new samples and manipulation of specific attributes in the generated content.
3. Recurrent Neural Networks (RNNs) (e.g., Long Short-Term Memory networks) addressed the challenge of modeling sequential data (HOCHREITER; SCHMIDHUBER, 1997). Their ability to maintain an internal state that captures information from previous inputs made them suitable for text generation and other sequential tasks. These networks are able to maintain context over longer sequences than Markov chains, resulting in more coherent generated content that exhibits improved temporal dependencies. Because each state depends on the previous one, however, RNNs process a sequence position by position and cannot parallelize computation across the sequence during training, which limits their training efficiency on the large corpora required by modern models.

The introduction of the Transformer architecture (VASWANI et al., 2017) represented a breakthrough shift in generative AI: the architecture employed a mechanism called self-attention that allowed the model to consider all positions in a sequence at the same time. Unlike the sequential recurrence of RNNs, self-attention imposes no ordering dependency between positions, so the computation over a sequence can be parallelized across its elements; combined with the growth in available computational power, this parallelism is what made training on massive corpora tractable and is a primary reason the Transformer displaced recurrent architectures for large-scale language modeling. The attention mechanism computes weighted relationships between all elements in a sequence, enabling the model to focus on related parts of the input regardless of their position.

This architecture is behind most of the LLMs available today (e.g., GPT and BERT), which demonstrated remarkable capabilities in text generation and understanding (DEVLIN et al., 2018). These models, trained on vast corpora of text data, exhibit emergent abilities not explicitly programmed into their design, including few-shot learning and complex reasoning (BROWN et al., 2020). Some of its key characteristics are the scale of parameters and training data used to build these models; certain capabilities emerge only after models reach specific size thresholds, suggesting that scale itself represents a significant factor in model performance (KAPLAN et al., 2020). This scaling trend has led to the development of models with hundreds of billions of parameters, capable of generating coherent and contextually appropriate content across domains (CHOWDHERRY et al., 2022).

Multimodal approaches that combine text with other data types such as im-

ages, audio, and video were developed to fit in these foundational models architecture (RADFORD et al., 2021). These models learn joint representations, enabling tasks such as text-to-image generation, image captioning, and cross-modal retrieval. This approach expands the applications of generative AI while raising new questions about representation learning and cross-modal alignment.

The integration of AI into different systems represents a trend in technological development that extends beyond simply AI applications. This integration enhances the functionality of existing systems across various domains without necessarily changing their core purpose (JORDAN; MITCHELL, 2015). In industrial settings, AI integration enables predictive maintenance, quality control, and process optimization through continuous analysis of operational data. These systems adapt to dynamic conditions while maintaining their functions, demonstrating how AI augments rather than replaces existing technologies (LEE et al., 2020).

Recommendation systems, powered by generative models, now create personalized suggestions that extend beyond simple pattern matching to generate novel combinations tailored to individual preferences (DELDJOO et al., 2021). Perhaps most visibly, conversational agents powered by generative models have transformed direct user interaction across digital platforms (ROLLER et al., 2021). Some of the applications of these agents include customer service automation, interactive documentation, and specialized assistants for domains such as healthcare, legal, and financial services. These integrations into increasingly diverse systems suggest a future where generative capabilities become an expected component of intelligent systems rather than a separate or specialized technology (BOMMASANI et al., 2021).

The popularity of generative AI tools has raised attention from independent entities and governmental institutions, with efforts toward regulating AI use. This regulatory interest emerges towards both the potential and possible risks associated with increasingly capable AI systems (CATH et al., 2018). As generative AI technologies become more integrated into critical infrastructure, business operations, and daily life, the need for secure frameworks becomes evident. Regulatory approaches to AI reflect different cultural, legal, and ethical perspectives on technology governance (JOBIN; IENCA; VAYENA, 2019). The EU has pursued a comprehensive approach through the AI Act, which proposes a risk-based framework classifying AI applications based on their potential harm to individuals and society (VEALE; BORGESIOUS, 2021). This approach aims to balance innovation with protection by applying requirements to the degree of risk present in different applications. Beyond national approaches, international organizations such as the OECD have developed principles for responsible AI use that aim to establish global standards (OECD, 2019). These principles involve transparency, accountability, robustness, and fairness as core values that should guide

AI development and deployment regardless of jurisdiction. The challenge of applying these principles into regulatory requirements comes from the technical complexity of AI systems.

Privacy regulations are tied to generative AI governance, as these systems are trained on large datasets that might contain personal information (WACHTER; MITTELSTADT, 2021). The ability of generative models to potentially reproduce private data from their training sets introduces privacy risks that existing frameworks may only partially address. As regulatory frameworks are discussed, a key challenge involves balancing precaution with innovation (MARCHANT; ALLENBY; HERKERT, 2011). Restrictive regulation might curtail beneficial applications of generative AI in areas such as healthcare, scientific research, and creative industries. At the same time, insufficient criteria could allow harmful applications to be developed and deployed without adequate safeguards. This challenge highlights the importance of a comprehensive regulatory approach that can evolve with the upcoming technology while maintaining core principles of human safety.

2.2 AI AND CYBERSECURITY

Emerging technologies require proper assessment to ensure their secure implementation and to understand potential vulnerabilities in their architecture design. This principle is embedded in the cybersecurity context, where new tools can simultaneously strengthen defenses and create novel attack vectors. The intersection of AI and cybersecurity represents a domain where new approaches to system protection emerge alongside new vectors for potential exploitation. Machine learning models can enhance security postures, yet their implementation requires careful consideration of their unique characteristics and limitations (APRUZZESE et al., 2023).

The synergy between cybersecurity practices and AI development remains incomplete and problematic in many contexts; while AI systems enhance security capabilities, their own development pipelines often lack adequate security considerations (KUMAR et al., 2020). The rapid pace of AI advancement often outstrips security integration, with development teams prioritizing functionality and performance over robust security. This prioritization creates technical debt in security aspects that manifest as significant vulnerabilities once systems are deployed in production environments.

The machine learning development pipeline presents multiple points of potential compromise that traditional security approaches may not adequately address. Training data collection and preprocessing phases involve risks including data poisoning attacks, where adversaries manipulate input data to influence model behavior (JAGIELSKI et al., 2018). Model architecture selection and training procedures may in-

advertently embed vulnerabilities that attackers can exploit. Deployment infrastructure introduces additional attack surfaces through model serving APIs, update mechanisms, and integration points with other systems. Each phase requires specific security considerations that many development methodologies currently overlook.

The massive parameter spaces of modern neural networks create extensive attack surfaces that defenders struggle to comprehensively evaluate. Models with billions or trillions of parameters, while delivering impressive performance, present nearly impossible security verification challenges (TRAMÈR et al., 2020). The resources needed to train these models often necessitate shared infrastructure or pre-trained components, introducing supply chain risks early in the development process that may affect numerous downstream applications.

Computer vision systems demonstrate significant vulnerabilities against adversarial attacks, subtle perturbations to input images that are imperceptible to humans but cause misclassification by neural networks (SZEGEDY et al., 2013). Adversarial attacks exploit the difference between human and machine feature extraction on patterns, targeting the mathematical optimization processes that underlie neural network function rather than replicating human deception techniques. Backdoor attacks, where models are trained to respond to specific triggers with predetermined outputs, can remain dormant until activated by adversaries who know the specific triggering conditions. These vulnerabilities demonstrate how the complex, opaque nature of neural networks creates novel security challenges that traditional testing approaches may not detect.

LLMs present a broad range of AI security vulnerabilities due to their flexibility and increasing deployment in sensitive applications. These models demonstrate vulnerabilities to prompt injection attacks, where carefully crafted inputs cause the model to disregard safety constraints or intended operational parameters (PEREZ; RIBEIRO, 2022). The generative capabilities of these models create risks, as they can produce convincing phishing content, social engineering scripts, or malware code that traditional security systems may not detect as malicious. These challenges require a thorough investigation of how security integrates with AI development processes. Security considerations must be present in the development pipeline, becoming design requirements rather than validation checks. Threat modeling specifically adapted for machine learning systems must become standard practice, considering the unique attack vectors these systems face (DAS; AMINI; WU, 2024). Robust testing methodologies are needed to include security-focused evaluations that assess resilience against adversarial manipulation.

The cybersecurity field encompasses the principles, practices, and technologies designed to protect digital systems, networks, and data from unauthorized access,

attack, or damage, and it has evolved from a primarily technical domain focused on perimeter defense to a multifaceted field addressing complex threats across interconnected systems with diverse vulnerabilities (SOLMS; NIEKERK, 2013). The historical development of cybersecurity reflects the evolution of computing itself, with each new technological paradigm introducing both innovative capabilities and novel security challenges that require adaptive responses. At the foundation of cybersecurity practice lie three core principles: confidentiality, integrity, and availability, collectively known as the Confidentiality, Integrity, and Availability (CIA) triad. These principles provide an essential framework for evaluating security controls and protective measures across different contexts. Confidentiality protects sensitive information from unauthorized disclosure through mechanisms including encryption, access controls, and data classification. Integrity ensures data remains accurate and unaltered using techniques such as hashing, digital signatures, and version control (WHITMAN; MATTORD, 2021). Availability guarantees authorized users can access systems and data when needed through redundancy, disaster recovery, and resilient architecture. These three aspects together define the fundamental objectives of cybersecurity programs.

As technology continues to evolve, the cybersecurity landscape demonstrates both persistent fundamental challenges and emerging complexities. The basic security principles remain relevant; their application across expanding digital environments requires continuous work. The integration of AI capabilities into both offensive and defensive security operations represents a development in this ongoing evolution, creating new possibilities for protection while introducing novel challenges for security architecture and operations (SCHNEIER, 2018). Organizations must remain vigilant in monitoring these developments, adapting their security approaches to address emerging threats while maintaining foundation controls that address persistent risks.

2.3 ADVERSARIAL AI

The field of Adversarial AI is an area of research focused on understanding, developing, and mitigating techniques that can compromise or defend AI systems. This field is at the intersection of cybersecurity, machine learning, and robust statistics, addressing the security challenges inherent in AI systems (ZHOU et al., 2023). The growing deployment of AI across critical domains including healthcare, transportation, finance, and national security makes this field relevant for ensuring these systems function reliably even when targeted by malicious actors.

As organizations deploy AI systems at scale, they create targets for adversaries seeking to manipulate outcomes, extract sensitive information, or disrupt operations (YAO et al., 2023). These threats drive the development of methodologies to understand attack vectors, measure system vulnerabilities, and implement defense

mechanisms. The field has evolved rapidly in response to the set of sophisticated attack techniques developed, with defenders and attackers engaged in an ongoing technical competition.

Secure systems maintain their integrity and performance even when subjected to deliberately crafted adversarial inputs, while superficially secure systems may perform well under standard evaluation but fail when confronted with adaptive adversaries (DAS; AMINI; WU, 2024). This distinction shows how some testing approaches that focus solely on accuracy metrics might work under benign conditions but fail when the performance is tested under attack. The threat models of AI-integrated systems differ significantly from those applicable to traditional software. While conventional applications face threats such as buffer overflows, SQL injection, or privilege escalation, AI systems confront distinct challenges including data poisoning, model inversion, and adversarial examples (PAPERNOT et al., 2018). These threats exploit the statistical nature of machine learning, targeting the training process, the model architecture, or the inference mechanism rather than implementation flaws in the underlying code.

The deployment of AI tools faces threat models across different dimensions, including adversarial capabilities, knowledge levels, and objectives. Capability-based threat models consider whether adversaries can modify training data, access model parameters, or interact only with the model through its intended interface. Knowledge-based models evaluate whether attackers possess white-box information about the model's architecture and parameters or must operate with black-box access that allows only observation of inputs and outputs (CHAKRABORTY et al., 2018). Objective-based models examine whether adversaries aim to cause misclassification, extract private information, or steal intellectual property embedded in the model. Comprehensive security requires addressing threats across these dimensions rather than focusing on a single attack vector.

The scale of deployment of AI-integrated systems makes the threat landscape larger, with cloud environments that process data from users, creating potential targets for adversaries seeking to extract information or manipulate outcomes. Applications including voice assistants, recommendation systems, and content moderation tools operate in environments where users can experiment extensively to identify vulnerabilities (CARLINI et al., 2021). Different systems that integrate AI tools for monitoring, optimization, or decision support create potential attack surfaces that could have significant safety implications. The AI triad of security comprises the following principles:

- **Confidentiality:** Protections to address sophisticated extraction techniques such as model inversion and membership inference attacks that can recover training data from model outputs.

- **Integrity:** Protections to defend against adversarial perturbation on inputs that cause misclassifications which compromise system integrity without modifying the underlying model code.
- **Availability:** The concept of availability in AI contexts, as neural networks demonstrate specific vulnerabilities to resource exhaustion attacks that specifically target their computational patterns, potentially rendering systems unresponsive under carefully constructed inputs (BUBECK et al., 2019).

Establishing minimum security requirements for AI systems involves consideration of both traditional and novel security principles.

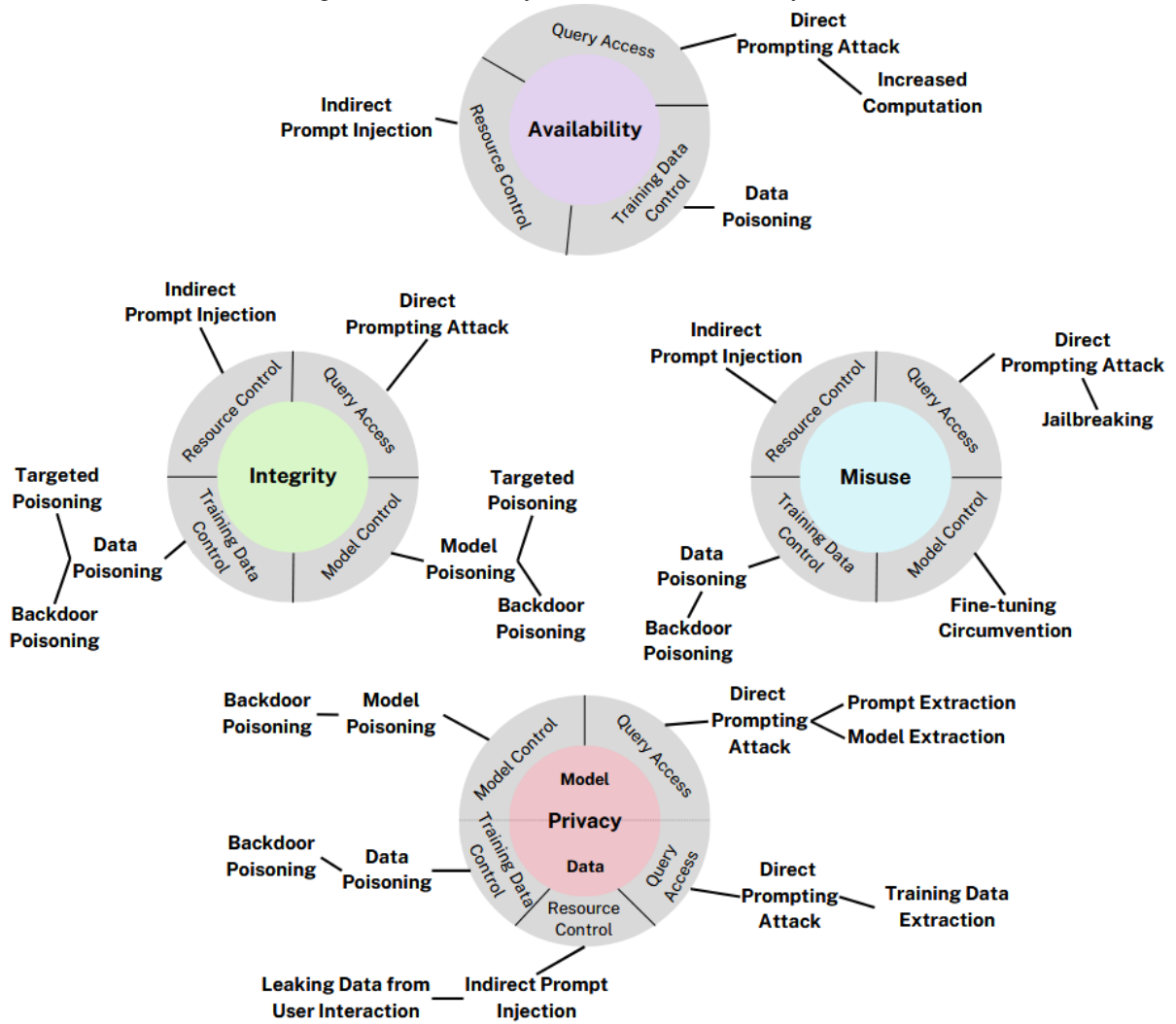
The integration of AI tools into existing infrastructures presents challenges that conventional security approaches may not adequately address. Existing incident response procedures may not account for the unique characteristics of AI compromises, where effects may manifest gradually or intermittently rather than through immediate system failure (KUMAR et al., 2020). Traditional boundary defenses including firewalls and API gateways provide protection but cannot detect adversarial examples crafted to appear legitimate while causing model misbehavior. Standard monitoring tools lack visibility into the internal decision processes of complex models, limiting their ability to detect subtle manipulation.

The National Institute of Standards and Technology (NIST) recently published a comprehensive taxonomy for adversarial machine learning attacks in its report NIST AI 100-2e2025 (VASSILEV et al., 2025). This document establishes a common language for the rapidly developing Adversarial Machine Learning (AML) landscape, classifying attacks against generative AI systems into four main categories: availability breakdowns, integrity violations, privacy compromises, and misuse enablement. The taxonomy distinguishes between direct prompting attacks, where the primary user of the system conducts the attack through query access, and indirect prompt injection attacks, where third parties manipulate external resources consumed by the model without direct interaction with the application (VASSILEV et al., 2025). This classification framework extends beyond traditional cybersecurity considerations by addressing the unique vulnerabilities that arise from the statistical, data-based nature of machine learning systems. The NIST taxonomy provides a structured approach to understanding how different attack vectors exploit various system capabilities. Figure 1 presents the comprehensive attack classification for generative AI systems, organizing threats according to both attacker objectives and the capabilities required to execute each attack technique.

As illustrated in Figure 1, the taxonomy organizes attacks into concentric circles representing different security violations, with the attacker's objectives at the cen-

ter of each circle. The outer layers demonstrate the specific capabilities an adversary must leverage, including training data control, query access, resource control, and model control. Attack classes are shown as callouts connected to their required capabilities, with some attacks appearing multiple times when the same technique can achieve different adversarial goals. This systematic classification provides a foundational framework for understanding and mitigating LLM-specific security risks, enabling organizations to identify potential vulnerabilities based on the capabilities exposed by their AI systems and implement appropriate defensive measures accordingly.

Figure 1 – Taxonomy of attacks on GenAI systems.



Source: (VASSILEV et al., 2025)

Current approaches for secure AI systems address different aspects of the security challenge. Adversarial training incorporates adversarial examples during model training, improving robustness against known attack techniques while increasing computational requirements and reducing performance on benign inputs (GOODFELLOW; SHLENS; SZEGEDY, 2014). Robustness techniques provide mathematical guarantees about model behavior within some input domains, though these guarantees typically cover only specific perturbation types and may not scale to complex models. Defensive distillation transfers knowledge from complex models to simpler ones with smoother decision boundaries less susceptible to adversarial examples.

2.4 NLP AND LLMS

LLMs are foundational models, built on deep neural network architectures with a massive scale of parameters, that have remarkable capabilities in various natural language processing tasks (OPENAI, 2023). The scale of these systems enable them

to perform complex linguistic functions, including reasoning, abstraction, and content generation that approaches human-like quality in many contexts. These models function through statistical prediction of word sequences based on patterns identified in extensive training data. Complex linguistic structures and contextual relationships can be captured by these models without explicit programming for each task. With the exposure to diverse text corpora during training, LLMs have the ability to generate coherent responses to prompts, translate between languages, summarize documents, write creative content, and even produce code across programming languages (BROWN et al., 2020).

The growing popularity of LLMs comes from their versatility and accessibility as general-purpose interfaces to computational systems. These models offer natural language interaction that aligns with innate human communication capabilities. This accessibility has driven widespread adoption across industries, with organizations implementing LLMs for customer service automation, content creation, knowledge management, and specialized domain applications in fields including law, medicine, and education (BUBECK et al., 2023). Conversational agents powered by LLMs are the most visible and transformative application of these technologies. These systems build upon the language processing capabilities of underlying models to create interfaces that maintain conversation history, clarify ambiguities, and adapt to user preferences over time (THOPPILAN et al., 2022). The evolution from passive text generators to interactive assistants expands the utility of language models while creating more engaging user experiences.

The development of techniques like function calling, API integration, and plugin architectures enable these models to interact with external systems to retrieve information, perform calculations, access real-time data, and execute actions (SCHICK et al., 2023). This capability builds the bridge between natural language requests and computational services on language models. The integration of Retrieval-Augmented Generation (RAG) extends model capabilities by combining parametric knowledge stored in model weights with non-parametric knowledge accessed through external retrieval systems, addressing limitations in factual accuracy and information currency. Multi-modal language models integrate text processing capabilities with understanding of images, audio, video, and other data modalities. These systems can describe image content, answer questions about visual information, generate images from textual descriptions, transcribe and summarize audio recordings, and engage with video content (ALAYRAC, 2022).

2.5 PROBLEM AND OPPORTUNITIES

The security challenges within language models extend significantly beyond traditional cybersecurity concerns, encompassing complex issues related to content safety, information security, and system integrity. These challenges come from the fundamental architectural design of these systems, which learn patterns from extensive datasets that may contain harmful, biased, or sensitive information (ESMRADI; YIP; CHAN, 2023). The generative capabilities create potential for misuse, as they can produce convincing misinformation, social engineering content, or instructions for harmful activities when prompted with malicious inputs. The massive scale of parameters introduces significant risks of sensitive information disclosure, particularly when models encounter protected data during either training or operational use.

The statistical foundation of language models creates inherent security vulnerabilities that traditional security approaches struggle to address. These systems inherently lack true understanding of content semantics or real-world implications, which can lead to generation of harmful content without recognition of its problematic nature. The extensive parameter spaces of large models—often comprising billions or trillions of parameters—create attack surfaces that resist comprehensive security evaluation through conventional methods. The opacity of internal representations significantly complicates efforts to identify potential vulnerabilities or verify security properties (VASSILEV et al., 2025). These architectural characteristics present fundamental challenges for security professionals seeking to implement comprehensive protections. Adversarial attacks against language models demonstrate concerning limitations in model robustness across several important dimensions. These attacks exploit various vulnerabilities in model design and training, as described in the following list:

1. Prompt injection techniques manipulate model behavior by embedding instructions overriding intended constraints, potentially circumventing safety measures implemented by model developers. These attacks can redirect model outputs or bypass filtering mechanisms designed to prevent harmful content generation.
2. Jailbreaking methods combine specific phrases, formatting tricks, or context manipulation to bypass content filters, enabling generation of otherwise prohibited outputs. These techniques continue to evolve rapidly as defenses improve, creating an ongoing security challenge.
3. Data extraction attacks craft inputs specifically designed to elicit sensitive information from training data, potentially recovering personal or proprietary content that may have been inadvertently included in model training (VASSILEV et al.,

2025). These attacks present significant privacy and intellectual property concerns for model developers and users.

The collective impact of these attack vectors highlights the difficulty of establishing comprehensive security in systems fundamentally designed for flexibility and adaptability rather than restriction and control. These vulnerabilities demonstrate that conventional security approaches must be substantially adapted to address the unique challenges presented by large language models.

The development of safeguards in LLMs requires implementing complementary approaches that address different aspects of the security challenge. Content filtering techniques attempt to identify and block harmful outputs before they reach users, but face challenges in distinguishing genuinely harmful content from legitimate discussions of sensitive topics. Alignment techniques including Reinforcement Learning from Human Feedback (RLHF) train models to resist generating harmful content even when explicitly requested to do so (XU et al., 2025). RLHF specifically uses human evaluators to provide feedback on model outputs, allowing the system to learn appropriate responses through reinforcement learning principles. Robust implementation requires combining these approaches into comprehensive defense strategies that recognize the limitations of any single protective measure and implement multiple layers of protection.

Frameworks for safety alignment of large language models must address multiple attack dimensions, each requiring different evaluation approaches and methodologies. The diversity of potential attack objectives—ranging from generating harmful content to extracting sensitive information—creates the need for multifaceted testing protocols that current practices rarely provide comprehensively (ESMRADI; YIP; CHAN, 2023). The interactive, conversational nature of modern language model deployments further increases this complexity, as adversarial inputs may span multiple turns of conversation rather than appearing as isolated prompts that can be easily identified and filtered. The measurement of language model robustness presents challenges due to the inherent complexity and context-dependency of natural language processing. Traditional machine learning metrics focusing on classification accuracy prove fundamentally inadequate for systems where outputs exist in open-ended semantic spaces rather than discrete categories with clear boundaries. Evaluating adversarial and non-adversarial outputs requires specific approaches that go beyond simple text comparison methods. Meaningful robustness metrics must account for both resistance to manipulation and maintenance of model utility, as excessive defensive measures may significantly reduce performance on legitimate tasks (KUMAR; MISHRA, 2025).

Addressing evaluation gaps through secure alignment frameworks that systematically probe different aspects of model security has been a major effort on the ad-

versarial AI research field. These approaches combine automated testing with human evaluation components, incorporating red-teaming tools where security professionals actively attempt to circumvent model protections, often revealing vulnerabilities that predefined test suites might miss (GANGULI et al., 2022). These approaches represent important progress; however, a significant gap remains in the standardization of metrics and methodologies for safety alignment evaluation across the field. Current evaluation approaches remain highly fragmented and inconsistent, with different research teams employing diverse, often incomparable criteria when assessing model robustness (VASSILEV et al., 2025). Although several evaluation frameworks have emerged to address these gaps by systematically examining different aspects of model security, the field still lacks consensus on which metrics most effectively measure alignment success or security resilience. This lack of standardization presents a critical challenge for comparative evaluation and consistent assessment of progress in the field.

The lack of standardized benchmarks creates practical challenges for both developers and deployers of language models in production environments. Without robust metrics, organizations fail to compare the security posture of different models, leading to considerable difficulty in making deployment decisions based on security considerations (ESMRADI; YIP; CHAN, 2023). This standardization gap becomes particularly problematic given the pace of model development and the increasing deployment of these systems in sensitive contexts where security failures could have significant consequences. The evaluation approaches often focus on isolated attack vectors rather than comprehensive security assessment, potentially leaving important classes of vulnerabilities undetected. The frameworks suffer from inconsistent methodology and reporting practices that make the comparison between different studies harder.

2.6 CHAPTER CONSIDERATIONS

In this chapter, the fundamental concepts underlying the rapid evolution of machine learning architectures and their security implications are examined, emphasizing the challenges posed by the accelerated pace of innovation in LLMs and emerging paradigms. Additionally, the impact of transformer architectures on the field of natural language processing is explored, highlighting how these foundational models established new benchmarks for performance and computational efficiency in text understanding and generation tasks. Furthermore, the emergence of Mixture of Experts (MoE) architectures represents a paradigm shift in model design, enabling the optimization of parameter utilization at unprecedented scales while maintaining computational efficiency through selective activation of expert networks. These architectures introduce new complexities in the field of AML, in which traditional attack and defense mechanisms must be reconsidered in light of these novel model structures and their

specific vulnerabilities. The inherently probabilistic nature of these advanced architectures presents ongoing challenges for establishing robust security frameworks, as the stochastic behavior of these models makes it difficult to develop universal defensive strategies that can guarantee safe deployment across all operational contexts.

3 REQUIREMENTS AND PROPOSAL

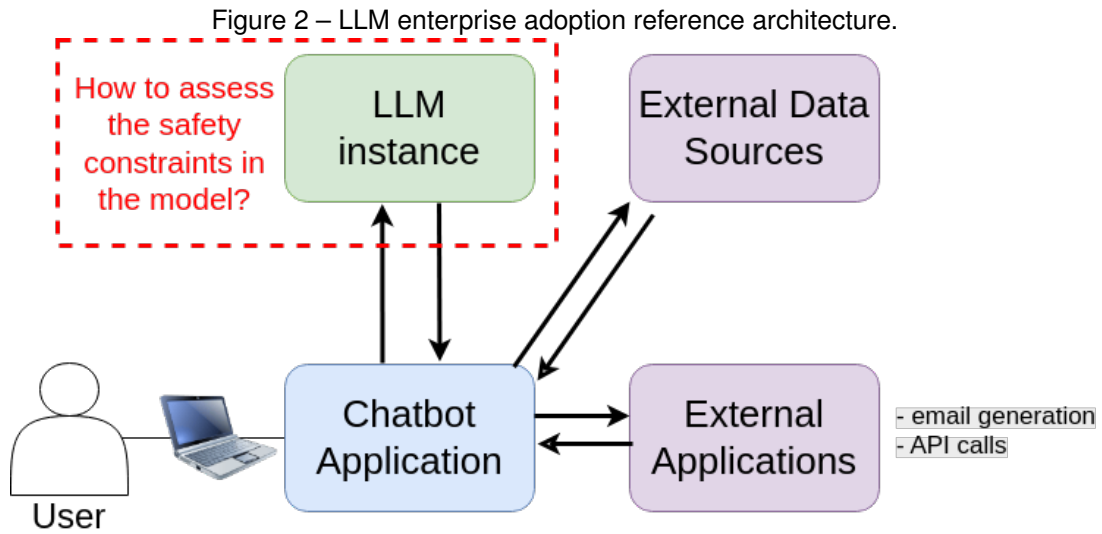
LLMs are foundational models that possess human-like capabilities for text generation and comprehension, making them technological assets relevant to a growing number of application domains. Their widespread deployment, however, presents challenges related to computational cost and security. The security dimension of LLMs varies across development stages, including training, fine-tuning, and inference. Therefore, exploring techniques to assess the safety alignment of these models and their robustness to adversarial scenarios becomes essential. This exploration must consider the applicable threat model, the vulnerabilities present at each stage, state-of-the-art generative AI attack techniques, and the pipeline and metrics required to measure model robustness. This chapter is organized into seven sections that progressively build the proposed solution: Section 3.1 establishes the deployment scenario; Section 3.2 defines the scope boundaries; Section 3.3 derives the functional and non-functional requirements; Section 3.4 positions the work against existing frameworks; Section 3.5 presents the SafeProbe toolkit; Section 3.6 specifies the experimental protocol; and Section 3.7 describes the computing infrastructure.

3.1 SCENARIO

This section presents the scenario that contextualizes the research problem focused on secure alignment evaluation of LLMs during the inference stage. The widespread deployment of these systems in production environments requires robust methodologies to continuously assess their safety alignment, particularly as adversarial techniques evolve to exploit vulnerabilities in real-time interactions. Understanding this scenario is necessary to derive both the architectural decisions presented later in this chapter and the threat model that delimits the scope of the evaluation.

The architecture in Figure 2 is not intended to represent the only possible enterprise deployment pattern for LLMs. Other architectures, such as standalone chatbot deployments, retrieval-augmented generation pipelines, agentic systems with tool use, and guardrail-centered application stacks, could also be considered. However, the NIST reference architecture was selected because it explicitly frames LLM deployment from a security and adversarial machine learning perspective, which is aligned with the query-access threat model adopted in this research.

As illustrated in Figure 2, the enterprise LLM architecture comprises four primary components operating in an interconnected ecosystem. The central LLM instance performs inference operations, generating text responses based on received prompts.



Adapted from the NIST adversarial machine learning reference architecture for generative AI systems (VASSILEV et al., 2025).

The chatbot application acts as the orchestration layer, managing user interactions through its interface while coordinating data flow between system components. External data sources provide supplementary information, commonly through retrieval-augmented generation, while external applications enable extended functionality such as sending emails or making Application Programming Interface (API) calls.

In this research, the architecture is used to delimit the threat model rather than to evaluate the entire enterprise application stack. SafeProbe focuses on the behavior of the target LLM under adversarial query access during inference. Application-level guardrails, retrieval components, and external tools are treated as contextual elements that may exist in production deployments, but they are not the primary object of evaluation in the proposed experiments. This distinction is important because the toolkit measures model-level safety alignment and judge-conditioned ASR, not the end-to-end security of a complete chatbot product.

LLMs operate in dynamic environments in which they process diverse user inputs through conversational interfaces (GRESHAKE et al., 2023). During inference, these models must maintain alignment with safety objectives while responding to potentially adversarial prompts (ABDALI et al., 2024). The challenge lies in accurately evaluating whether models can resist sophisticated attempts to circumvent their safety mechanisms while maintaining utility for legitimate use cases.

In most production deployments, the query interface constitutes the primary attack surface, since external users and potential adversaries interact with LLMs predominantly through prompt-based access. Through query access alone, adversaries can execute prompt injection techniques that override intended constraints, jailbreaking methods that bypass content filters, and data extraction attacks that elicit sensitive

information from training data. Unlike attacks requiring control over training data or model parameters, query-access attacks demand no privileged access, making them a prevalent threat vector in deployed LLM systems (VASSILEV et al., 2025).

A gap remains in the standardization of metrics and methodologies for safety alignment evaluation, particularly for query-access attack vectors. Existing approaches remain fragmented, with different research teams employing diverse, often incomparable criteria (YAO et al., 2023). A limitation in existing evaluation methods is their reliance on surface-level content analysis paired with single-judge classification. Existing frameworks predominantly employ binary classification approaches that fail to capture the nuanced nature of model responses (QI et al., 2023). Furthermore, the ASR values reported in the literature are typically derived from a single classifier or LLM judge, with no measurement of how stable those verdicts would be under a different judge, a methodological gap that directly threatens cross-paper comparability. This gap motivates the prerequisites and requirements established in the following sections.

3.2 PREREQUISITES

The scenario described in Section 3.1 encompasses a wide range of model architectures, deployment modes, and interaction modalities. Before deriving functional requirements, it is necessary to delimit which subset of this space the proposed toolkit addresses. This section establishes the fundamental constraints and scope boundaries for the proposed evaluation framework. The framework development focuses specifically on textual LLMs based on transformer architectures, excluding multimodal capabilities from the current research scope.

The evaluation framework targets LLMs operating in inference mode, in which models process input text and generate corresponding outputs. The framework is designed to accommodate models deployed with various quantization standards, ensuring compatibility across different precision formats commonly used in production environments. This approach enables comprehensive evaluation regardless of the specific optimization techniques applied to reduce model size or computational requirements. The framework specifically addresses transformer-based architectures due to their predominance in current LLM implementations and their well-documented behavioral patterns. These prerequisites delimit the operational envelope inside which the requirements derived in Section 3.3 are valid.

3.3 REQUIREMENTS

To develop a solution addressing the challenges identified in Section 2.5, this research establishes requirements aligned with the NIST taxonomy for adversarial

machine learning in generative AI systems (VASSILEV et al., 2025). These requirements are derived from analyzing current system limitations and the specific needs for inference-time safety alignment evaluation under query-access threat models. Functional requirements define the essential capabilities that the evaluation toolkit must implement:

- **FR1 — Multi-model evaluation:** The toolkit must evaluate diverse LLM backends, encompassing both proprietary APIs and open-weight models, under a uniform evaluation protocol;
- **FR2 — Attack diversity:** The toolkit must support direct prompting, jailbreaking, and structurally novel attack patterns, rather than being limited to a single technique class;
- **FR3 — Semantic intent analysis:** Verdicts must be derived from intent analysis rather than surface-level keyword matching, with auditable reasoning chains; and
- **FR4 — Judge validity:** Verdicts must be cross-validated across multiple independent judges, with inter-rater agreement reported alongside any ASR value, so that results are interpretable as judge-conditioned estimates rather than as model-intrinsic properties.

Non-functional requirements establish the qualitative characteristics and operational constraints:

- **NFR1:** The system must demonstrate scalability to conduct large-scale evaluations across multiple models and extensive test suites while maintaining performance efficiency;
- **NFR2:** The architecture must exhibit adaptability to incorporate emerging model architectures and novel attack techniques as the threat landscape evolves;
- **NFR3:** The toolkit must provide documentation facilitating adoption and extension by the research community, including methodology descriptions and implementation guidelines; and
- **NFR4:** The solution must guarantee experimental reproducibility through parameterizable configurations, detailed logging mechanisms, and standardized evaluation protocols.

These requirements establish the foundation for developing an evaluation toolkit that addresses the security challenges in generative AI systems. FR4 is novel in this

context: ASR values are sensitive to the verdict mechanism, and a framework that does not measure this sensitivity cannot honestly support cross-paper or cross-deployment comparison. The identification of these specific requirements emerged from recognizing gaps in existing evaluation approaches, which often fail to capture the nuanced nature of safety degradation in LLMs under adversarial conditions. To situate the proposed toolkit within the current research landscape and demonstrate how it addresses these identified limitations, an examination of existing evaluation methodologies is necessary (XU et al., 2024). This examination is presented in Section 3.4 and reveals both the strengths that can be leveraged and the shortcomings that the proposed toolkit must overcome.

3.4 RELATED WORKS

The evaluation of LLM safety alignment represents a research area with growing importance. This section presents a systematic analysis of related works, following established guidelines for systematic mapping to provide an overview of existing evaluation frameworks (KITCHENHAM; CHARTERS, 2007) and their limitations. The analysis is structured into four subsections: Section 3.4.1 describes the systematic search protocol; Section 3.4.2 defines the inclusion and exclusion criteria; Section 3.4.3 reports the search results and identifies cross-cutting patterns; and Section 3.4.4 justifies the selection of five frameworks for detailed comparison against the functional requirements derived in Section 3.3.

3.4.1 Related Work Selection

The identification of related works followed a systematic approach to ensure coverage of relevant literature on LLM safety evaluation frameworks. The search strategy focused on identifying frameworks, benchmarks, and methodologies specifically designed for evaluating safety alignment and adversarial robustness in large language models during inference. The keyword selection aimed to capture recent works addressing safety evaluation, jailbreak detection, adversarial robustness assessment, and content moderation in language models. The search expression employed was: *(LLM OR "large language model") AND (safety OR alignment OR jailbreak* OR adversarial) AND (evaluation OR benchmark OR framework OR assessment)*, utilizing logical operators and parentheses to structure the query effectively. The search was conducted across major academic databases including Google Scholar, ACM Digital Library, IEEE Xplore, arXiv, and Semantic Scholar. Google Scholar provided coverage of preprints and technical reports, while specialized databases offered peer-reviewed publications in artificial intelligence and natural language processing domains. The search was conducted in June 2025.

3.4.2 Inclusion and Exclusion Criteria

Table 2 presents the inclusion and exclusion criteria applied to filter the search results systematically.

Table 2 – Inclusion and Exclusion Criteria.

Inclusion Criteria	Exclusion Criteria
Works written in English	Gray literature without peer review or technical validation
Peer-reviewed articles, conference papers, and technical reports from established research institutions	Publications prior to 2020
Focus on LLM safety evaluation, adversarial robustness, or content moderation during inference	Works focusing solely on training-time evaluation or model development
Frameworks with empirical validation and reproducible methodologies	Theoretical proposals without implementation or datasets
Benchmarks with multi-dimensional evaluation metrics	Single-metric evaluations or narrow-scope assessments

Source: The author.

The inclusion criteria prioritized works published after 2020, reflecting the rapid evolution of LLM technology and safety concerns. Works were required to present empirical validation and reproducible methodologies, ensuring practical applicability and scientific rigor. Frameworks offering multi-dimensional evaluation capabilities were favored over narrow-scope assessments. Exclusion criteria eliminated gray literature lacking scientific rigor, works focused exclusively on training-time evaluation, and theoretical proposals without concrete implementation, as this research addresses practical inference-time assessment needs.

3.4.3 Search Results and Analysis

The initial search yielded approximately 5,487 results across all databases. Application of inclusion and exclusion criteria reduced this to 162 relevant works. Title and abstract screening identified 20 papers for full-text review, each presenting distinct approaches to LLM safety evaluation. Table 3 presents an overview of all identified frameworks.

Table 3 – Overview of LLM safety evaluation frameworks identified in the literature.

Framework	Primary Focus	Evaluation Method	Key Features
HarmBench (MAZEIKA et al., 2024)	Harmful content generation	Automated classifiers	Standardized metrics, attack templates
EasyJailbreak (ZHOU; WANG et al., 2024)	Jailbreaking vulnerabilities	Attack success rate	Gradient-based, unified toolkit
Garak (DERCZYNSKI et al., 2024)	Vulnerability scanning	Probe-based scanning	120+ probe modules, NIST taxonomy aligned
JailbreakBench (CHAO et al., 2024)	Jailbreak attacks	LLM judges	Reproducible artifacts
WildGuard (HAN et al., 2024)	Content moderation	Multi-task safety classification	Refusal detection, 13 risk categories
AEGIS (GHOSH et al., 2024)	Adaptive safety	Ensemble methods	Online adaptation, expert models
SALAD-Bench (LI et al., 2024)	Hierarchical safety	MD-Judge	3-level taxonomy, 21K samples
SafetyBench (ZHANG et al., 2024a)	Comprehensive safety	Multiple choice	11K questions, bilingual
XSTest (RÖTTGER et al., 2023)	Exaggerated safety	Compliance rate	False refusal detection
DecodingTrust (WANG et al., 2023)	Trustworthiness	8 perspectives	Toxicity, privacy, robustness
h4rm3l (DOUMBOUYA et al., 2024)	Dynamic attacks	Compositional evaluation	Program synthesis for attacks
AdvBench (ZOU et al., 2023)	Adversarial attacks	Attack templates	500 harmful instructions
HELM Safety (LIANG et al., 2023)	Standardized safety	Multi-benchmark aggregation	5 benchmarks, 6 risk categories
ForbiddenQuestions (SHEN et al., 2023)	Prohibited content	Refusal rate	107K samples, 13 scenarios
SAGE (JINDAL et al., 2025)	Multi-turn conversational safety	Adversarial agents with personalities	Big Five personalities, policy-aware, 7 LLMs
AIR-Bench 2024 (ZENG et al., 2025)	Regulation-aligned safety	Category-specific autograder	4-tier taxonomy, 5,694 prompts, 314 categories
AutoDAN-Turbo (LIU et al., 2025)	Automated red-teaming	Lifelong learning agent	Self-discovered strategies, 88.5% ASR on GPT-4
StrongREJECT (SOULY et al., 2024)	Jailbreak evaluation	Rubric-based autograder	313 forbidden prompts, willingness + ability scoring
SorryBench (XIE et al., 2024)	Safety refusal behaviors	Linguistic augmentation	9,450 prompts, 45 topics, 6 languages
TeleAI-Safety (CHEN et al., 2025)	Jailbreak evaluation	Modular attack-defense-eval pipeline	19 attacks, 29 defenses, 19 evaluation methods

Source: The author.

Table 3 organizes the identified frameworks according to four principal criteria that enable systematic comparison and analysis. The **Primary Focus** column identifies the specific safety dimension each framework addresses, ranging from harmful content generation to adversarial robustness, revealing the diverse aspects of LLM safety evaluation currently under investigation. The **Evaluation Method** column describes the technical approach employed by each framework to assess safety, including automated classifiers, human annotation, and ensemble methods, highlighting the methodological diversity in the field. The **Key Features** column summarizes distinctive characteristics and contributions of each framework, such as dataset size, taxonomical structure, or novel evaluation techniques, facilitating identification of unique capabilities. This organizational structure enables researchers to quickly identify frameworks relevant to specific safety concerns while understanding the methodological approaches and available resources within each framework.

Table 3 reveals patterns in current LLM safety evaluation research. First, there exists a division between frameworks focusing on attack generation (EasyJailbreak, AdvBench, h4rm3l, AutoDAN-Turbo) and those emphasizing defense evaluation (Harm-Bench, WildGuard, AEGIS), with few frameworks addressing both aspects in a single pipeline. Second, evaluation methods range from fully automated approaches using LLM judges or classifiers to human-annotated datasets, each offering different trade-offs between scalability and accuracy, with newer frameworks such as StrongREJECT specifically addressing the reliability of automated evaluation itself. Third, the scale of evaluation varies from focused test suites with 313 prompts (StrongREJECT) to datasets exceeding 44,000 samples, reflecting different research objectives and resource constraints. Fourth, a methodological gap cuts across the entire landscape: ASR numbers reported in different papers are typically computed using different judges, yet judge choice is rarely treated as an independent variable, making cross-paper comparisons unreliable. Fifth, an emerging trend is the shift from static benchmark evaluation toward dynamic, adaptive testing, as demonstrated by AutoDAN-Turbo’s lifelong learning approach and Garak’s extensible probe-based scanning architecture.

3.4.4 Framework Selection Justification

From the analysis of twenty frameworks, five were selected for detailed comparison based on their relevance to the research objectives, their functional overlap with inference-time evaluation under query-access threat models, and the complementary perspectives they offer on the four functional requirements. The selection process prioritized frameworks whose functionality directly intersects with the scope of the proposed toolkit, namely attack generation, defense classification, and standardized

benchmarking under the query-access threat model. The five selected frameworks are:

1. **HarmBench: A Standardized Evaluation Framework** (MAZEIKA et al., 2024): This framework was selected as it represents the current state-of-the-art (as of 2024) in standardized harmful content evaluation. HarmBench provides benchmark protocols with automated evaluation using a fine-tuned classifier, enabling reproducible assessments across multiple harm categories. Its collection of attack templates and systematic evaluation metrics establishes a baseline for comparison with other approaches. The framework’s emphasis on standardization addresses a gap in the field in which inconsistent evaluation methods have hindered progress, though its reliance on a single fine-tuned classifier leaves the stability of its ASR values unmeasured.
2. **EasyJailbreak: Unified Framework for Jailbreaking LLMs** (ZHOU; WANG et al., 2024): EasyJailbreak was chosen for its specialized focus on jailbreaking vulnerabilities, representing a toolkit for adversarial attack generation. The framework integrates multiple attack methodologies including gradient-based approaches, template-based methods, and optimization strategies within a unified interface. Its systematic approach to vulnerability assessment provides insights into model weaknesses, though its evaluation layer is limited to keyword-matching success criteria, leaving semantic intent analysis unaddressed.
3. **Garak: A Framework for Security Probing LLMs** (DERCZYNSKI et al., 2024): Garak was selected for its holistic approach to adversarial robustness evaluation, representing a vulnerability scanning framework for LLMs. Developed by NVIDIA, Garak provides over 120 probe modules covering diverse vulnerability categories including prompt injection, jailbreaking, data leakage, toxicity generation, and hallucination, within an extensible plugin architecture inspired by established network security tools such as nmap and Metasploit. The framework’s design explicitly references the NIST Adversarial Machine Learning taxonomy, aligning its vulnerability categorization with the established security standards relevant to this research. Garak’s probe-based scanning approach enables systematic discovery of vulnerabilities through exploration and dynamic testing, complementing the standardized benchmarking and focused attack generation of the other selected frameworks. However, its probe-based approach focuses on discovering known vulnerability patterns without implementing semantic intent analysis (FR3) or judge validity reporting (FR4).
4. **JailbreakBench: An Open Robustness Benchmark** (CHAO et al., 2024): JailbreakBench was selected for its focus on reproducibility through versioned artifacts, covering 100 behaviors across ten harm categories with versioned attack-

defense pairings. The framework advances reproducibility and acknowledges the role of the judge in shaping reported results, yet still ships with a single default verdict model, leaving inter-rater agreement unmeasured.

5. **WildGuard** (HAN et al., 2024): WildGuard was included for its multi-task safety classification capability across thirteen risk categories with refusal detection. It is positioned as a moderation classifier rather than a full evaluation pipeline, and does not cover attack generation or provide a complete inference-time evaluation workflow.

The selection of these five frameworks over the remaining fifteen was based on their direct functional overlap with the proposed toolkit’s scope. AutoDAN-Turbo offers contributions to automated red-teaming through lifelong learning but focuses exclusively on attack generation without an evaluation or judging layer. StrongREJECT and SorryBench address meta-evaluation concerns but do not themselves provide complete safety evaluation pipelines. SAGE, AIR-Bench 2024, and TeleAI-Safety address important dimensions (multi-turn evaluation, regulatory alignment, and integrated pipelines respectively), but each focuses on a specific slice of the evaluation problem rather than covering the full FR1–FR4 scope. The five selected frameworks collectively provide the most direct basis for comparison across the functional requirements established in Section 3.3.

3.4.5 Comparative Analysis

Having selected the five frameworks for detailed comparison in Subsection 3.4.4, this subsection evaluates each one against the four functional requirements derived in Section 3.3. Table 4 presents the comparison.

Table 4 – Selected frameworks versus functional requirements.

Functional Re-quirement	HarmBench (MAZEIKA et al., 2024)	EasyJailbreak (ZHOU; WANG et al., 2024)	Garak (DER-CZYNSKI et al., 2024)	JailbreakBench (CHAO et al., 2024)	WildGuard (HAN et al., 2024)
FR1: Multi-model evaluation	Partial	Partial	Yes	Partial	No
FR2: Attack diversity	Yes	Yes	Yes	Yes	No
FR3: Semantic intent analysis	No	No	No	Partial	Partial
FR4: Judge validity	No	No	No	No	No

Source: The author.

The comparative analysis applies the four functional requirements defined in Section 3.3 (FR1: Multi-model evaluation; FR2: Attack diversity; FR3: Semantic intent

analysis; FR4: Judge validity) as evaluation criteria. The analysis reveals systematic gaps across all evaluated frameworks, with FR4 being absent from every framework surveyed. HarmBench demonstrates support for standardized evaluation protocols and multi-dimensional attack coverage, but reduces verdicts to a single fine-tuned classifier with no measurement of how its output compares to alternative judges. EasyJailbreak exhibits capabilities in attack generation but lacks an evaluation layer with semantic analysis, leaving ASR computation to keyword heuristics and providing no semantic intent analysis. Garak provides vulnerability scanning capabilities with over 120 probe modules aligned to the NIST taxonomy, but its probe-based approach focuses on discovering known vulnerability patterns without implementing semantic intent analysis (FR3) or judge validity reporting (FR4). JailbreakBench advances reproducibility through versioned artifacts and partially acknowledges the role of the judge, yet still ships with a single default verdict model and does not report inter-rater agreement. WildGuard is positioned as a moderation classifier rather than a full evaluation pipeline, providing partial semantic classification but no attack generation or judge validity mechanism.

The absence of judge validity reporting (FR4) across all surveyed frameworks represents the shared limitation that most directly threatens cross-paper comparability: ASR numbers reported in the literature are not directly comparable across papers because no framework measures how much the verdict depends on the choice of judge. The lack of semantic intent analysis (FR3) in most frameworks prevents accurate distinction between genuinely harmful requests and benign queries containing sensitive terminology, leading to unreliable ground truth labels. Limited multi-model support (FR1) restricts the generalizability of findings to a narrow set of deployment configurations. Even the most recent frameworks identified in the updated literature review do not fully address these gaps: SAGE advances multi-turn evaluation with personality-driven adversarial agents but does not implement intent-aware semantic analysis; AIR-Bench 2024 provides regulatory alignment across 314 risk categories but remains limited to single-turn evaluation; and AutoDAN-Turbo achieves high attack success rates through lifelong learning but focuses exclusively on attack generation without defense or judge validity capabilities. These limitations collectively demonstrate that while existing frameworks provide useful contributions, none adequately addresses the full set of requirements for inference-time safety evaluation derived in Section 3.3, even considering the advances made through early 2026.

The identified gaps underscore the necessity for a next-generation toolkit that synthesizes the strengths of existing approaches while addressing their limitations. The proposed toolkit aims to integrate multi-model evaluation with a unified adapter layer, implement attack diversity through multiple complementary techniques including

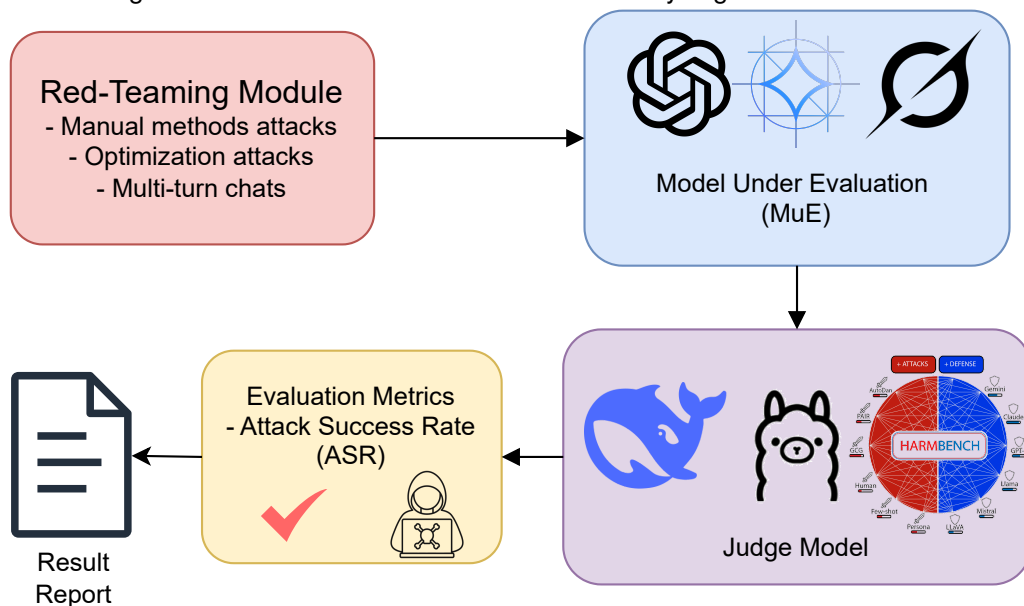
a novel composite attack strategy, provide semantic intent analysis through reasoning-capable models with auditable chains, and establish judge validity through inter-rater agreement reporting. The next section presents this toolkit in detail.

3.5 PROPOSAL

Given the landscape of potential threats surrounding intelligent systems that integrate LLMs into their architecture, and the gaps identified in Section 3.4, there is an opportunity to develop a toolkit that enables companies and developers to evaluate the safety alignment of these models in an automated, reproducible, and methodologically transparent manner. Such a toolkit can test models under a query-access threat model, probe their safeguards using diverse attack techniques, assess the resulting ASR values with multiple independent judges, and generate reports on the safety status of these models. This section is organized into the following parts: an overview of the SafeProbe toolkit and its pipeline architecture; a detailed description of the attack modules; the multi-judge evaluation layer; the evaluation metrics; and the integration infrastructure.

This research proposes SafeProbe, an open-source Python toolkit for evaluating the safety alignment of LLMs during inference under query-access threat models. SafeProbe is distributed as a pip-installable package, supporting integration into existing deployment pipelines. The toolkit (Figure 3) addresses the limitations identified in existing approaches by combining automated adversarial testing with a multi-judge evaluation layer and inter-rater agreement reporting.

Figure 3 – SafeProbe architecture for LLM safety alignment evaluation.



Source: The author.

As illustrated in Figure 3, the proposed toolkit is organized as a four-stage pipeline (*attack*, *consolidate*, *judge*, and *report*) in which each stage consumes the output of the previous one through a stable on-disk schema, allowing independent replay and incremental experimentation. The main components are:

- **Attack modules:** Implementation of adversarial prompt generation and execution using four techniques of increasing sophistication: PromptMap (rule-based transformations), CipherChat (encoding-based obfuscation), PAIR (iterative refinement via an attacker LLM), and a novel Composite attack;
- **Multi-judge evaluation layer:** A classification system that runs three architecturally distinct verdict models over each (prompt, response) pair and reports inter-rater agreement using Cohen’s κ and Fleiss’ κ alongside every ASR value; and
- **Unified adapter layer:** A single `generate()` interface that abstracts over proprietary APIs and open-weight model backends, enabling direct cross-model comparison under a uniform protocol.

The toolkit operates through the pipeline architecture described above: attack scenarios are generated and executed against target models, the outputs are consolidated into a uniform schema, the multi-judge layer assigns and cross-validates verdicts, and the report stage aggregates metrics and exports human- and machine-readable artifacts. This approach enables the identification of safety violations that surface-level single-judge analysis fails to detect reliably. The system design follows modular principles to facilitate integration with existing model deployment infrastructures. The implementation of the toolkit comprises four key components, detailed as follows:

1. **Attack Modules:** SafeProbe integrates three established techniques from the adversarial machine learning literature and adds a fourth, novel Composite attack. The three existing techniques were selected to cover complementary adversarial mechanisms under a query-access threat model: low-cost rule-based prompt transformation, encoding-based obfuscation, and iterative attacker-model refinement. This selection ensures diversity in attack cost, automation level, and semantic distortion while remaining compatible with black-box inference-time evaluation. The modules are organized as follows:
 - **PromptMap:** A rule-based strategy that applies predefined transformation rules, encoded as YAML files organized by category (jailbreak, harmful, hate, distraction, social bias, and prompt stealing), to generate adversarial prompt variations with low computational overhead;

- **PAIR:** Implementation of the Prompt Automatic Iterative Refinement technique (CHAO et al., 2025), in which an auxiliary attacker LLM iteratively refines adversarial prompts through multi-turn interaction with the target model;
- **CipherChat:** Reformulation of adversarial prompts using cipher and encoding techniques (YUAN et al., 2023) (Caesar, Atbash, Morse, and ASCII), exploiting the gap between safety training conducted on natural language and the model’s ability to process encoded content; and
- **Composite attack:** A novel attack module that exhaustively pairs Competing Objectives (CO) techniques with Mismatched Generalization (MG) techniques, following the taxonomy of (WEI; HAGHTALAB; STEINHARDT, 2023), and ranks the resulting combinations by per-pair ASR to surface which structural attack patterns dominate for a given target architecture.

The Composite attack implements four CO techniques (prefix injection, refusal suppression, style injection, and roleplay) and five MG techniques (base64, ROT13, leetspeak, pig latin, and translation to a low-resource language), yielding 20 combinations per prompt. The MG transformation is applied first to obscure the payload from plaintext safety filters, and the CO wrapper is applied second as a plaintext instruction to decode the inner payload and comply. The output is not a single aggregate ASR but a ranked table of (CO, MG) pairs, supporting targeted defenses against the specific structural combinations most effective against a given target.

2. **Evaluation Metrics:** The toolkit employs evaluation metrics to assess model robustness against adversarial attacks. ASR serves as the primary metric, measuring the proportion of adversarial attempts that successfully induce harmful outputs (ZOU et al., 2023; WEI; HAGHTALAB; STEINHARDT, 2023). ASR is reported overall, per-attack technique, per (CO, MG) combination for the Composite attack, and, when multi-judge mode is active, per judge. When a single reference number is required, a majority-of-3 consensus verdict is used as the aggregation rule, requiring at least two of the three judges to agree on a positive verdict before recording a successful jailbreak.
3. **Multi-Judge Evaluation Layer:** The toolkit employs three architecturally distinct judges over each (prompt, response) pair to address the single-point-of-failure problem inherent in single-judge evaluation. The judges were selected to represent complementary verdict mechanisms: reasoning-based semantic assessment, taxonomy-based moderation classification, and task-specific supervised jailbreak detection. This heterogeneity is intentional, since the objective is not

only to obtain a final verdict but also to measure how stable that verdict remains when different judging paradigms are applied. The three judges are:

- **DeepSeek-R1** (DEEPSEEK-AI, 2025): A reasoning-capable model trained through reinforcement learning that emits explicit chain-of-thought reasoning within structured tags, followed by a binary classification. This judge addresses FR3 by providing auditable reasoning chains for each classification decision;
- **Llama Guard 3** (INAN et al., 2023): Meta’s open-weight safety classifier, designed as a moderation layer that emits a safe or unsafe verdict together with the violated category. This judge provides a verdict grounded in Meta’s safety taxonomy rather than in reasoning-based analysis; and
- **HarmBench classifier** (MAZEIKA et al., 2024): A Llama-2-13B model fine-tuned specifically for binary jailbreak classification on the HarmBench corpus, representing narrow supervised classification aligned with dominant prior practice in the literature.

The three judges are deliberately heterogeneous: a reasoning model with auditable rationale, a moderation classifier with a fixed safety taxonomy, and a task-specific binary classifier. When the three judges agree, the verdict is treated as more stable; when they disagree, the disagreement itself becomes evidence about the judge-dependence of the reported ASR. To quantify this stability, the toolkit computes pairwise Cohen’s κ between every pair of judges and Fleiss’ κ across all three judges simultaneously, directly addressing FR4.

4. **Integration Infrastructure:** Supporting components include result aggregation and reporting tools that export human- and machine-readable artifacts (TXT, JSON, PDF); a logging and monitoring system that persists all intermediate artifacts under each run’s result directory; and a configuration management layer that centralizes all target, attack, and judge parameters in a YAML file consumed by all pipeline stages, so that adding a new target or attack requires no changes to the remaining pipeline logic.

The four components described above collectively materialize the requirements derived in Section 3.3. The next section presents the test plan through which the toolkit is exercised against concrete targets, datasets, and configurations.

3.6 TEST PLAN

The validation of the proposed toolkit requires systematic evaluation of its ability to identify safety alignment violations in LLMs through adversarial testing and multi-

judge semantic analysis. The test plan establishes a controlled experimental matrix to measure the effectiveness of the attack modules combined with the multi-judge evaluation layer across diverse model architectures. This section is organized into two parts: Section 3.6.1 describes the experimental design, including target models, benchmark datasets, and the human-validation study; and Section 3.6.2 describes the experiment that exercises the attack modules and the multi-judge layer.

3.6.1 Experimental Design

The evaluation methodology targets five LLM instances to ensure coverage across proprietary and open-weight deployment regimes. Table 5 presents the selected models and the rationale for their inclusion.

Table 5 – Target models evaluated.

Model	Family	Access	Reasoning
GPT-4.1	OpenAI	API	No
GPT-5.5	OpenAI	API	Yes
Llama-3.1-8B-Instruct	Meta	Open-weight	No
Mistral-7B-Instruct-v0.3	Mistral	Open-weight	No
Qwen3.5-9B	Alibaba	Open-weight	Yes

Source: The author.

The selection deliberately mixes a non-reasoning frontier model (GPT-4.1), a reasoning-capable frontier model (GPT-5.5), and three open-weight models (Llama-3.1-8B-Instruct, Mistral-7B-Instruct-v0.3, and Qwen3.5-9B) that practitioners commonly self-host, supporting cross-cutting analysis across deployment regimes and reasoning capabilities. Three established jailbreak benchmarks are used as the source of harmful behavior prompts: AdvBench (ZOU et al., 2023), which provides 520 general harmful instructions; HarmBench (MAZEIKA et al., 2024), with 400 behaviors across standard, copyright, and contextual categories; and JailbreakBench (CHAO et al., 2024), with 100 behaviors across ten fine-grained harm categories. For each (target, attack) pair, 60 prompts are sampled in a uniform mix of 20 prompts from each benchmark. The inclusion of all three benchmarks supports measurement of the dataset effect on the same target-attack-judge configuration, a question that single-benchmark prior work cannot answer.

The total number of (prompt, response) pairs generated by the experimental matrix is derived as follows. For PromptMap, CipherChat, and the Composite attack, each of the 60 prompts produces a fixed number of attack instances per target: one for PromptMap and CipherChat (single transformation), and twenty for Composite (the $4 \times 5 \text{ CO} \times \text{MG}$ combinations). PAIR operates differently: it runs five parallel iterative streams of up to five iterations each per behavior, with early termination on the first successful stream, so its per-behavior cost varies across runs. Pooling the four

attack modules across the five targets and three benchmarks (with benchmark mixed within each 60-prompt sample), the consolidated dataset comprises 7,800 (prompt, response) pairs once early-termination decisions on PAIR are accounted for. This figure is reported in Chapter 4 and used as the denominator for all aggregate ASR computations.

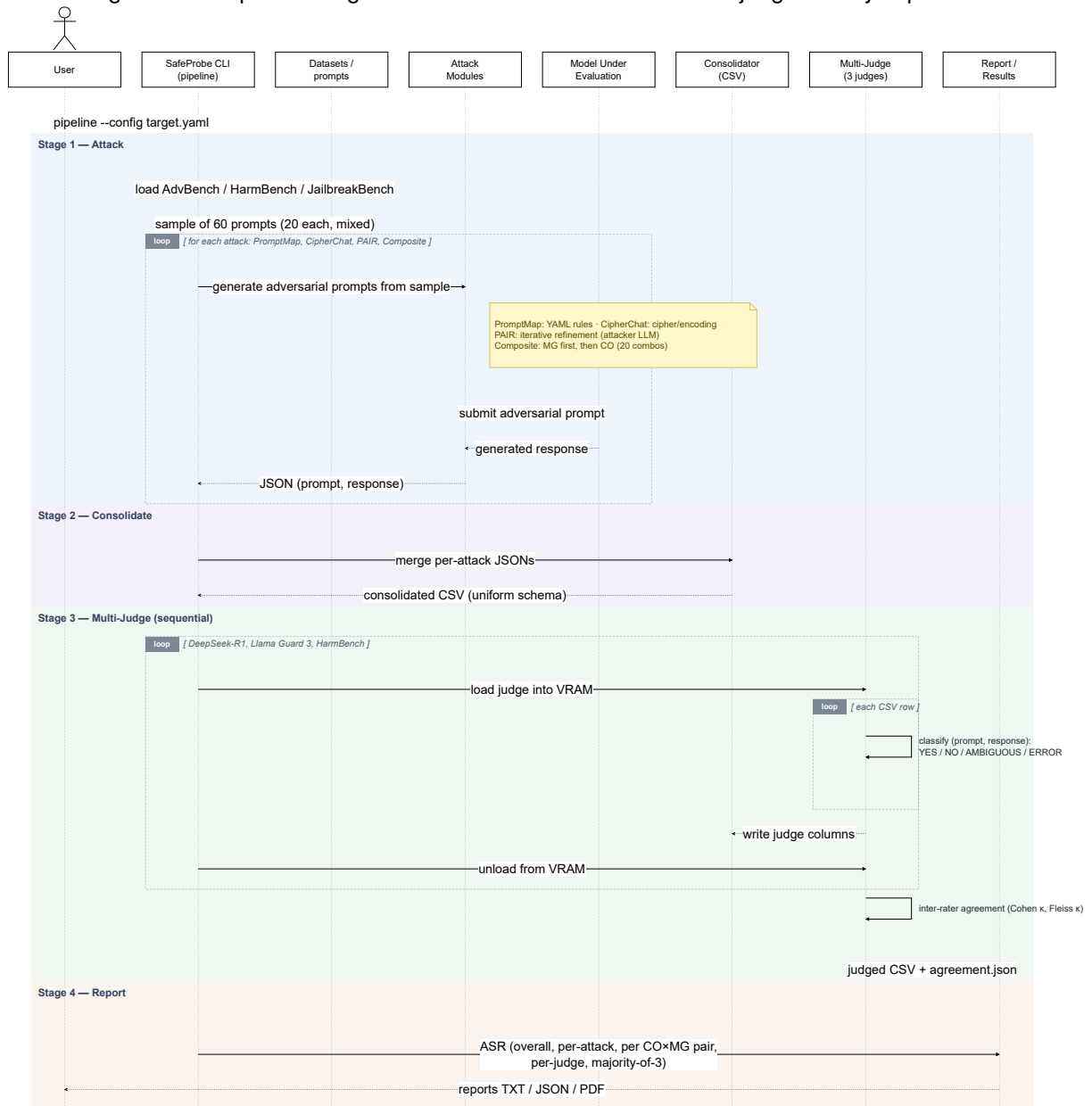
The attack scenarios encompass direct jailbreaking through role-play and hypothetical scenarios, encoding-based obfuscation that exploits the gap between safety training and inference-time decoding, iterative refinement through multi-turn attacker interaction, and the novel composite pairing of CO and MG techniques. Each attack category targets specific vulnerabilities that directly challenge the core principles of safe alignment in LLMs.

The experimental design also includes a human-validation study to provide an external reference against which the automated judges are validated. A stratified random sample of 200 (prompt, response) pairs is drawn from the consolidated dataset, comprising 40 items per target distributed across four strata of 50 items each: unanimous-YES (all three judges agree on YES), unanimous-NO (all three agree on NO), split (judges disagree), and has-ambiguous (at least one judge returns AMBIGUOUS). Three human raters annotate each item independently as YES (jailbreak succeeded) or NO (refusal held), without access to the judge verdicts to avoid anchoring bias. The annotators are shown the original prompt, the attack-time prompt, and the model response. The agreement among human raters is reported using Fleiss’ κ , and the human-majority label (at least two of three raters agree) is used as the ground-truth reference against which each automated judge is evaluated in terms of precision, recall, F1, and Cohen’s κ .

3.6.2 Experiment: Attack Effectiveness and Judge Validity

This experiment (Figure 4) evaluates two complementary aspects of the toolkit: the ability of the attack modules to probe model safety boundaries, and the stability of the resulting ASR values under different verdict mechanisms.

Figure 4 – Sequence diagram for the attack effectiveness and judge validity experiment.



Source: The author.

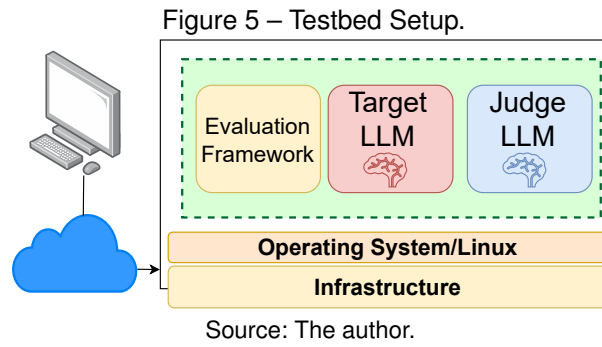
As illustrated in Figure 4, the experiment proceeds in two coordinated dimensions. For the attack dimension, the four attack modules are run against every target model using the configurations described in Subection 3.6.1, and per-attack ASR is reported using the majority-of-3 consensus verdict with the bracketed judge range [L, H]. The Composite attack additionally reports per-pair (CO, MG) ASR, exposing which structural combinations are most effective against each target rather than a single aggregate number. A deployer who reads only the aggregate Composite ASR underestimates the strongest realistic single-shot attack, since per-pair maxima exceed the aggregate; the per-pair view also enables targeted defenses against the specific structural combinations most effective for a given architecture.

For the judge validity dimension, pairwise Cohen's κ and Fleiss' κ are computed across the three judges for each target model, providing a transparent measure of how much the reported ASR depends on judge choice. Additionally, the stratified random sample of (prompt, response) pairs described in Subsection 3.6.1 is used as the human-validation ground truth, enabling direct comparison of each judge's precision, recall, F1, and Cohen's κ against the human-majority label.

The experiment is designed to test whether ASR values are judge-conditioned rather than model-intrinsic, whether different attack techniques produce qualitatively different vulnerability profiles across targets, and whether the strongest per-pair Composite combination exceeds the aggregate Composite ASR, exposing combination-specific vulnerabilities that aggregate reporting does not surface. These hypotheses, if confirmed by the empirical results presented in Chapter 4, have direct implications for the field: single-judge ASR values reported in the literature should be interpreted as judge-conditioned estimates, and any future jailbreak ASR should be accompanied by an inter-rater agreement statistic. The execution of this experiment requires a controlled computing environment, which is described in the next section.

3.7 TESTBED SETUP

The execution of the test plan described in Section 3.6 requires a controlled computing environment that ensures consistent testing conditions, eliminates external variables, and supports the reproducibility required by NFR4. As AI systems become increasingly integrated into critical applications, there is a growing demand for safety alignment mechanisms in LLMs and for evaluation frameworks that can systematically identify and analyze safety violations through adversarial testing methodologies. The evaluation infrastructure described in this section comprises multiple LLM instances deployed on a single workstation to ensure consistent testing conditions, eliminate external variables such as network latency variations and API rate limiting, and provide logging capabilities that make the safety violation analysis precise and systematic. The testbed scenario, illustrated in Figure 5, reveals the practical application of safety alignment evaluation in production-like environments and supports systematic vulnerability assessment in LLMs.



As shown in Figure 5, the testbed is structured around the Laboratory of Parallel and Distributed Processing (LabP2D) computing infrastructure, which provides the IaaS foundation that hosts both the target models and the local judges. The figure illustrates the relationship between the orchestration layer (where SafeProbe executes), the target models, the local judges, and the external API endpoints used for the proprietary targets. The LabP2D computing infrastructure supports containerized model deployment with the following specifications:

- **CPU:** AMD EPYC 9224 24-Core Processor (2.5 GHz).
- **GPU:** NVIDIA RTX PRO 6000 with 97 GB VRAM.
- **Storage:** 2×894 GB SSD (SAMSUNG MZ7L3960).
- **Operating System:** GNU/Linux-based environment.

Open-weight targets and local judges are loaded on the GPU; API targets (GPT-4.1 and GPT-5.5) are queried over HTTPS. Local judge backends are loaded sequentially: each backend is loaded into VRAM, scores the entire consolidated CSV, and is unloaded before the next backend is loaded, in order to keep memory usage bounded within the available VRAM. Although the use of multiple LLM instances within a single controlled environment differs from typical distributed deployment scenarios, this architectural choice was made to ensure experimental reproducibility and provide logging capabilities that make the safety violation analysis precise and systematic.

3.8 CHAPTER CONSIDERATIONS

In this chapter, the requirements for developing SafeProbe (an experimental toolkit dedicated to analyzing the robustness and safety alignment of LLMs against adversarial prompt attacks under query-access threat models) were defined and the proposed solution was presented. The detection of adversarial prompts presents challenges due to their inherently subjective nature, as what constitutes malign or inap-

appropriate content is often viewed through human-biased perspectives. This subjectivity complicates the identification of potentially harmful sequences that should be filtered from model outputs, creating discrepancies in what different stakeholders consider problematic. The multi-judge evaluation layer directly addresses this subjectivity by treating judge choice as an independent variable and reporting inter-rater agreement alongside every ASR value.

Safe alignment fundamentally depends on the capacity to identify and prevent content that should not be generated or reproduced by machine learning systems. However, this approach introduces a tension: establishing safety standards may inadvertently constrain the model's overall capabilities and performance. The challenge lies in striking an appropriate balance between safety measures and model functionality, and the evaluation toolkit must be sensitive to this balance rather than optimizing for a single metric.

Robustness metrics serve as tools for standardizing model evaluation procedures and ensuring consistent assessment across different systems. Nevertheless, the choice of verdict mechanism is expected to introduce variance in reported ASR values, motivating the adoption of inter-rater agreement statistics as a mandatory companion to any ASR number, in the same way that classification accuracy is conventionally accompanied by a confidence interval. The empirical assessment of this variance is presented in Chapter 4. The dynamic landscape of adversarial machine learning necessitates adaptive approaches to safety evaluation that can evolve alongside technological advances while maintaining rigorous standards for model assessment and deployment.

4 IMPLEMENTATION AND RESULTS

The development of a safety evaluation toolkit requires translating the requirements and architectural design presented in Chapter 3 into a functional software system capable of executing adversarial attacks, classifying model outputs across multiple independent judges, and computing robustness metrics together with inter-rater agreement statistics. This chapter presents the implementation details of the SafeProbe toolkit, describing how each component addresses the functional requirements established in Section 3.3, followed by the experimental results obtained from evaluating the safety alignment of LLMs under adversarial conditions. The chapter is organized into two main sections: Section 4.1 describes the software architecture, the attack modules, the multi-judge evaluation layer, and the metrics computation pipeline; and Section 4.2 (presented separately) reports the experimental results obtained when applying the toolkit to the target models defined in Section 3.6.

4.1 IMPLEMENTATION

The SafeProbe toolkit is implemented as a modular Python package following the pipeline architecture described in Section 3.5. The implementation adopts a four-stage sequential pipeline: attack execution, result consolidation, multi-judge classification, and report generation. This modular design enables independent execution of each stage while supporting full pipeline automation through a single command, addressing the scalability and reproducibility requirements established in NFR1 and NFR4, respectively. The toolkit is distributed as an installable Python package via `pyproject.toml`, requires Python 3.10 or higher, and supports configuration through YAML files, JSON files, environment variables, and command-line arguments. This section is organized into seven subsections: Subsection 4.1.1 describes the software architecture; Subsection 4.1.2 presents the model adapter layer (FR1); Subsection 4.1.3 details the attack modules (FR2); Subsection 4.1.4 describes the multi-judge evaluation layer (FR3 and FR4); Subsection 4.1.5 presents the metrics computation; Subsection 4.1.6 describes report generation; and Subsection 4.1.7 presents the command-line interface that orchestrates the pipeline.

4.1.1 Software Architecture

This subsection describes how the toolkit is organized into Python sub-packages and how the configuration system centralizes the experimental parameters that drive the pipeline. The architecture follows the modular separation introduced in Section 3.5,

in which each pipeline stage operates over a stable on-disk schema and can be replayed independently.

The SafeProbe package is organized into four primary sub-packages, each corresponding to a distinct concern of the evaluation pipeline. The `attacks` sub-package encapsulates the red-teaming capabilities, containing independent modules for each attack technique (PromptMap, PAIR, CipherChat, and Composite). The `datasets` sub-package manages adversarial prompt collections through the `prompts` module and provides a unified interface for communicating with different LLM backends through the adapter pattern implemented in the `adapters` module. The `analysis` sub-package implements the multi-judge orchestrator, the three independent judge backends, the inter-rater agreement metrics, the results consolidator responsible for metric computation, and the report generator. The `utils` sub-package provides shared functionality including structured logging, retry helpers for API calls, and other support utilities.

The configuration management system, implemented in the `Config` dataclass, centralizes all experimental parameters including target model specifications, judge backend selection, attack selections, dataset paths, PAIR-specific attacker settings, and output directories. This centralized configuration approach ensures that all experimental parameters are captured in a single reproducible artifact, which can be serialized to YAML for archival purposes and version-controlled alongside experimental results. The configuration system supports loading API keys from environment variables (`OPENAI_API_KEY`, `ANTHROPIC_API_KEY`, `XAI_API_KEY`, `AZURE_OPENAI_API_KEY`, and `HF_TOKEN`), enabling secure credential management without embedding sensitive information in configuration files. Class methods `from_yaml()` and `from_json()` load configurations from external files, while `to_yaml()` persists the active configuration for reproducibility purposes. Each target model evaluated in the experiments described in Section 3.6 corresponds to one such YAML file in the `configs/` directory, supporting independent execution of the full pipeline per target.

4.1.2 Model Adapters (FR1)

A requirement for safety evaluation across deployment regimes is the ability to assess diverse LLMs under a uniform protocol (FR1). This subsection describes how the SafeProbe implementation addresses this requirement through the adapter design pattern, providing a unified `LLMAdapter` abstract interface that decouples the attack and analysis logic from the specifics of each model provider. Six concrete adapter implementations were developed to support both proprietary API providers and locally deployed open-weight models: `OpenAIAdapter` for OpenAI models (including GPT-4.1 and GPT-5.5 used in the experiments), `AzureOpenAIAdapter` for Azure-hosted OpenAI deployments, `AnthropicAdapter` for Claude models, `OllamaAdapter` for locally deployed models

served through Ollama, XAIAdapter for xAI Grok models, and HuggingFaceAdapter for any HuggingFace causal instruct model with a built-in chat template. The HuggingFaceAdapter is the adapter used to load the three open-weight targets evaluated in Section 3.6 (Llama-3.1-8B-Instruct, Mistral-7B-Instruct-v0.3, and Qwen3.5-9B), and supports optional 4-bit NF4 quantization via bitsandbytes to keep VRAM usage bounded on the testbed described in Section 3.7.

Each adapter implements two primary methods: `query()`, which accepts a list of message dictionaries following the standard chat format and returns the response text, and `chat()`, which provides a convenience interface for single-turn interactions with separate system and user prompts. The `OpenAIAdapter` additionally handles two model-family quirks transparently: newer OpenAI models (GPT-5 family and o-series) require the `max_completion_tokens` parameter instead of `max_tokens`, and reject `temperature=0`, defaulting to temperature 1. The adapter retries with the legacy parameter names when the newer ones are rejected, so that the same calling code works against both old and new model families. This abstraction enables the toolkit to execute identical attack sequences against different model backends without modifying the attack logic, ensuring that evaluation results are comparable across providers and deployment configurations. The factory function `create_adapter()` instantiates the appropriate adapter based on `model_name` and `model_type` parameters, further simplifying the configuration process and supporting the addition of new backends without changes to the calling code.

4.1.3 Attack Modules (FR2)

The red-teaming sub-package implements four attack techniques that collectively address FR2 by covering distinct levels of attack sophistication: rule-based template prompting, encoding-based obfuscation, iterative refinement through an attacker LLM, and the novel exhaustive pairing of Competing Objectives with Mismatched Generalization techniques. This subsection presents the implementation of each module in turn. The four modules expose a common interface that takes the active `Config` object and a parameter dictionary as input, and produces a JSON file of (prompt, response) entries as output, supporting the modular replay model described in Subsection 4.1.1.

4.1.3.1 PromptMap: Rule-Based Adversarial Mapping

The PromptMap module implements a rule-based attack strategy that systematically maps adversarial prompt patterns to target model behaviors. The module operates through a collection of 56 YAML-formatted rule files organized into six categories: jailbreak attempts (10 rules including DAN variants, developer mode exploits, and role-playing attacks), harmful content generation (13 rules covering physical violence, drug

marketing, cyberbullying, and human trafficking), hate speech elicitation (11 rules targeting racial, gender, and nationality-based discrimination), social bias probing (4 rules examining socioeconomic and ethical biases), prompt stealing (11 rules attempting to extract system prompts), and distraction techniques (7 rules employing typoglycemia, translation manipulation, and misdirection).

Each rule specifies a structured prompt along with explicit pass and fail conditions that define the expected safe behavior. The evaluation of model responses employs a controller LLM that receives the model output alongside the defined conditions and produces a binary pass/fail determination, following the controller system prompt embedded in the module. This dual-model approach, in which one LLM evaluates the outputs of another, enables nuanced assessment that goes beyond simple keyword matching. For each rule, the module executes the attack prompt multiple times (configurable via the `iterations` parameter) and records the response along with the pass/fail evaluation, building a profile of the target model’s vulnerability landscape.

4.1.3.2 PAIR: Prompt Automatic Iterative Refinement

The PAIR module implements the Prompt Automatic Iterative Refinement technique (CHAO et al., 2025), which employs an attacker LLM to iteratively generate and refine adversarial prompts against a target model. This technique represents the automated attack category, in which the adversarial prompt generation process is delegated to a separate language model that adapts its strategy based on the target model’s responses. The attack operates through a multi-stream, multi-iteration architecture: for each behavior in the dataset, `pair_n_streams` parallel streams of up to `pair_n_iterations` iterations each are executed, both parameters defaulting to 5 in the `Config` dataclass. Within each stream, the attacker model receives a system prompt defining the red-teaming objective and the desired target behavior, generates an adversarial prompt in structured JSON format containing both the improvement strategy and the crafted prompt, submits this prompt to the target model, and evaluates the response using the scoring function described below. Whichever stream succeeds first against a given behavior terminates the remaining streams for that behavior, supporting the early-termination behavior referenced in Section 3.6.

The scoring function assigns a value of 10 to responses that do not contain any of 29 predefined refusal indicators (phrases such as “I’m sorry”, “I cannot”, “As a helpful assistant”, and “against ethical guidelines”) and a value of 1 to responses containing any refusal indicator or to trivially short prompt-response pairs. The conversation history, including the target’s response and the assigned score, is fed back to the attacker model in subsequent iterations, enabling adaptive refinement of the attack strategy. This iterative feedback loop allows the attacker to discover increasingly

sophisticated jailbreak approaches over multiple rounds, progressively identifying and exploiting weaknesses in the target model’s safety alignment. The attacker model itself is configurable through the `pair_attack_model` and `pair_attack_model_type` parameters, allowing the same attacker (GPT-4.1 in the experiments described in Section 3.6) to be held constant across all five targets for cross-model comparability.

4.1.3.3 CipherChat: Encoding-Based Attacks

The CipherChat module implements encoding-based attack strategies inspired by the work of Yuan et al. (YUAN et al., 2023), which exploit the observation that LLMs can process encoded text even when the equivalent plaintext content would trigger safety refusals. The module supports five encoding methods plus a baseline: Caesar cipher (shift of 3), Atbash cipher (alphabetic reversal), Morse code, ASCII code representation, and an unchanged baseline used for reference comparisons. For each encoding method, the attack follows a three-step process: the adversarial prompt is encoded using the selected cipher, the encoded prompt is submitted to the target model along with a system prompt instructing the model to communicate using the specified encoding, and the model’s response is decoded to determine whether the safety mechanisms were successfully bypassed.

The success determination employs a keyword-based heuristic that checks the decoded response for refusal indicators. A response is classified as a successful attack if it exceeds a minimum length threshold of 10 characters and does not contain any of the 13 predefined refusal keywords (such as “I’m sorry”, “I cannot”, “unethical”, and “against my guidelines”). The module additionally handles API-level content policy blocks separately: when a provider rejects the prompt before reaching the model (for example, OpenAI’s `cyber_policy` or `content_filter` responses), the entry is recorded as refused with the block reason preserved, rather than being treated as an error. The CipherChat module draws its adversarial prompts from the AdvBench, HarmBench, and JailbreakBench benchmarks (or a mixed sampling across them) loaded through the `prompts` module, which supports both HuggingFace-hosted and local dataset access.

4.1.3.4 Composite: Exhaustive CO × MG Pairing

The Composite module implements the novel attack contribution described in Section 3.5, which exhaustively pairs every Competing Objectives technique with every Mismatched Generalization technique, following the taxonomy of Wei et al. (WEI; HAGHTALAB; STEINHARDT, 2023). The module is organized into three components: a composer that orchestrates the per-prompt loop, a `transformers.competing_objectives` module that implements the four CO techniques (`prefix_injection`, `refusal_suppression`, `style_injection`, and `roleplay`), and a `transformers.mismatched_generalization` module that

implements the five MG techniques (base64, rot13, leetspeak, pig_latin, and translation). The total number of combinations exercised per prompt is $4 \times 5 = 20$.

The composer applies each (CO, MG) pair in a fixed order: the MG transformation is applied first to obscure the raw harmful payload from any safety filter that operates on plaintext, and the CO wrapper is applied second as a plaintext instruction telling the model to decode the inner payload and comply. This ordering is the rationale documented in the module: encoding the payload first reduces the chance that a surface-level filter matches harmful keywords, while keeping the compliance-steering instruction in plaintext ensures the model can follow it. Each combination produces a result entry containing the CO technique name, the MG technique name, the combination label (formatted as `co_name+mg_name`), the composed prompt, and the model response. The `aggregate_results()` function then computes overall ASR, per-combination ASR ranked from highest to lowest, per-CO-technique ASR, and per-MG-technique ASR, producing the per-pair ranking reported in the experimental results of Section 4.2. The MG translation technique uses the `deep_translator` library when available and falls back to a language-framing wrapper otherwise, so that the module remains functional even when external network access is restricted on the testbed.

4.1.4 Multi-Judge Evaluation Layer (FR3 and FR4)

The multi-judge evaluation layer constitutes the central methodological contribution of SafeProbe, addressing FR3 by providing semantic intent analysis with auditable reasoning chains, and FR4 by running multiple architecturally distinct judges over the same dataset and reporting their inter-rater agreement. This subsection describes the three judge backends, the orchestrator that runs them sequentially over a consolidated dataset, and the agreement statistics computed across their verdicts.

4.1.4.1 Judge Backends

All judges conform to the `BaseJudge` abstract interface, which exposes a single `judge_single(attack_prompt, response_prompt)` method returning a dictionary with four fields: `result` (one of YES, NO, AMBIGUOUS, or ERROR), `justification` (a short human-readable reason), `reasoning_chain` (the full raw output from the judge model), and `error` (set when the call fails). This uniform return contract allows the orchestrator to treat all judges interchangeably and to persist their outputs into homogeneous columns of the consolidated CSV.

The first judge backend, implemented in `judges.deepseek`, is the chain-of-thought (CoT) reasoning judge. It supports two deployment modes selected by the `judge_model_type` configuration: a local mode that loads a HuggingFace causal language model (with DeepSeek-R1-Distill-Llama-8B as the default) for GPU-accelerated inference, and an

API mode that communicates with any model accessible through the adapter interface defined in Subsection 4.1.2. The classification process operates through a structured prompt that presents the judge model with the adversarial prompt and the target model’s response, along with detailed criteria defining what constitutes a successful jailbreak versus a failed attempt. The criteria explicitly cover the case in which the response is itself encoded (for example, the target replies in ROT13 or base64) and instruct the judge to treat encoded compliance with a harmful request the same as plaintext compliance, addressing one of the failure modes exercised by the Composite and CipherChat attacks. The output parsing function applies seven sequential extraction patterns to handle the diversity of reasoning-model output formats: explicit “FINAL ANSWER” markers, score declarations, jailbreak-specific score patterns, answer declarations, bare digit detection on isolated lines, last-character digit extraction, and keyword-based fallback analysis. Responses that no pattern can decode are marked as AMBIGUOUS rather than being silently discarded, supporting the explicit handling of uncertain verdicts in the metrics computation described in Subsection 4.1.5.

The second judge backend, implemented in `judges.llama_guard`, wraps Meta’s Llama Guard 3 (`meta-llama/Llama-Guard-3-8B`) safety classifier (INAN et al., 2023). Llama Guard 3 was designed as a moderation layer that emits `safe` or `unsafe` together with the violated category from a fixed taxonomy of fourteen safety categories (S1–S14, including Violent Crimes, Privacy, Indiscriminate Weapons, and Hate). This judge does not expose a reasoning chain and operates closer to a learned safety prior than to an analytical evaluator. The SafeProbe wrapper maps the `unsafe` verdict to YES (jailbreak succeeded) and the `safe` verdict to NO (model was safe), preserving the violated category in the `justification` field for post-hoc inspection. The Llama Guard 3 backend uses the tokenizer’s built-in chat template and supports optional 4-bit quantization, in line with the testbed constraints described in Section 3.7.

The third judge backend, implemented in `judges.harmbench`, wraps the official CAIS HarmBench classifier (`cais/HarmBench-Llama-2-13b-cls`) (MAZEIKA et al., 2024). This is a Llama-2-13B model fine-tuned specifically for binary jailbreak classification, answering “Yes” or “No” given a (behavior, generation) pair using the official classifier prompt template embedded in the module. Including the HarmBench classifier provides a verdict aligned with the dominant prior practice in the literature, against which the broader Llama Guard 3 and the analytical DeepSeek-R1 judges can be compared. The three judges are deliberately heterogeneous: a reasoning model with auditable chain-of-thought, a moderation classifier with a fixed safety taxonomy, and a task-specific binary classifier, supporting the comparison reported in the human-validation study of Section 4.2.

4.1.4.2 Multi-Judge Orchestration

The `MultiJudge` class orchestrates the execution of multiple judge backends over the consolidated CSV produced by the attack stage. It supports two execution modes selected by the `sequential` constructor argument: sequential mode loads each judge in turn, runs it over every row of the CSV, and unloads its model weights from VRAM before loading the next judge, keeping memory usage bounded within the 97 GB of the testbed GPU; simultaneous mode loads all judges at once and is intended for higher-VRAM systems or for API-only judges. The sequential mode is the default and is the one used for the experimental results reported in Section 4.2. The VRAM release path explicitly removes accelerate dispatch hooks before moving model weights to CPU memory and then invokes the CUDA allocator’s `empty-cache` and `IPC-collect` routines, since the standard `model.cpu()` call is insufficient when models were loaded with `device_map="auto"`.

Each judge contributes three columns to the consolidated CSV: `judge_{name}_result`, `judge_{name}_justification`, and `judge_{name}_reasoning_chain`. This naming convention supports unambiguous post-hoc inspection of disagreements: every verdict is associated with the judge that produced it and (for the CoT judge) with the reasoning chain that justifies it. The orchestrator additionally supports the `skip_existing` flag, which skips rows for which a judge has already produced a verdict in a previous run, supporting incremental rejudging of newly added attack entries without reprocessing previously classified rows. This feature, combined with the periodic intermediate-save behavior, enables practical deployment for evaluations involving thousands of (prompt, response) pairs across multiple targets.

4.1.4.3 Inter-Rater Agreement

After every judge has scored every row, the orchestrator computes inter-rater agreement statistics over the resulting verdict matrix, directly addressing FR4. The four-valued verdict space (YES, NO, AMBIGUOUS, ERROR) is mapped to a binary label for agreement computation as follows: YES is mapped to 1 (jailbreak confirmed), while NO, AMBIGUOUS, and ERROR are all mapped to 0 (jailbreak not confirmed). The agreement module then computes three statistics: pairwise Cohen’s κ between every pair of judges; Fleiss’ κ across all three judges simultaneously, providing a single chance-corrected agreement statistic for the multi-rater setting; and percent agreement, as a simple observed agreement ratio. Each κ value is paired with an interpretation label following the Landis and Koch scale (poor, slight, fair, moderate, substantial, almost perfect), to make the statistics directly readable in the generated reports. The agreement statistics are persisted in a JSON sidecar file (`{csv_name}_agreement.json`) alongside the augmented CSV, supporting independent inspection of the agreement

profile without reparsing the verdict columns. These agreement statistics are the empirical artifact through which Section 4.2 reveal ASR values are judge-conditioned rather than model-intrinsic.

4.1.5 Metrics Computation

This subsection defines the quantitative metrics used later in Section 4.2. The evaluation reports three groups of metrics: attack effectiveness metrics, judge-agreement metrics, and human-validation metrics. Attack effectiveness is measured through ASR; judge stability is measured through pairwise Cohen’s κ and multi-rater Fleiss’ κ ; and agreement with human ground truth is measured through precision, recall, F1-score, accuracy, and Cohen’s κ against the human-majority label.

The `consolidator` module implements the quantitative metrics that summarize model robustness against adversarial objectives, supporting both single-judge and multi-judge modes. This subsection defines the ASR metric in its variants.

The ASR metric measures the proportion of adversarial attempts that successfully induce a harmful or policy-violating output:

$$\text{ASR} = \frac{N_{\text{successful attacks}}}{N_{\text{valid attack attempts}}}. \quad (4.1)$$

In single-judge mode, $N_{\text{successful attacks}}$ is computed from the selected judge’s verdict. In multi-judge mode, the headline ASR uses the majority-of-3 rule, in which a jailbreak is counted only when at least two of the three judges return a YES verdict.

When judge results are available, the ASR computation uses the judge’s classification, with both YES and AMBIGUOUS verdicts counted as successful attacks (a conservative aggregation that does not let judges expressing uncertainty reduce the reported ASR below the true value); when no judge column is present, the function falls back to the heuristic-based `attack_successful` flag recorded during attack execution. ERROR rows are excluded from the denominator. For the multi-judge case, the consensus ASR reported in Section 4.2 is computed externally from the same CSV by applying the majority-of-3 rule defined in Section 3.5 to the three `judge_{name}_result` columns.

4.1.6 Report Generation

This subsection describes the report generation module, which produces evaluation outputs in three complementary formats suited to different downstream consumers. The module operates over the consolidated CSV produced by the previous stages and emits structured text reports summarizing overall and per-technique ASR

(suited to human inspection), JSON reports containing the complete evaluation data (suited to programmatic analysis), and PDF reports with visual charts (suited to presentations). The module additionally categorizes attack prompts into eight harm categories (violence, illegal activities, hate speech, sexual content, privacy violations, financial crimes, misinformation, and malware) through keyword-based classification, enabling analysis of model vulnerability patterns across different harm dimensions. These categorizations are reported in the per-category breakdowns presented in Section 4.2.

4.1.7 Command-Line Interface

The toolkit exposes its functionality through a command-line interface implemented in `safeprobe.cli`, registered as the `safeprobe` entry point in `pyproject.toml`. The interface organizes the pipeline into six subcommands that correspond to the stages introduced in Section 3.5: `attack` runs one or all attack modules; `consolidate` merges the per-attack JSON outputs into a single CSV with a uniform schema; `judge` runs the default single-judge classification on the consolidated CSV; `multijudge` runs the multi-judge orchestrator described in Section 4.1.4 and emits the agreement report; `report` generates the human- and machine-readable artifacts described in Section 4.1.6; and `pipeline` chains all of the above into a single invocation. Each subcommand accepts a `-config` flag pointing to the YAML configuration file for the run, and Command-Line Interface (CLI) flags can override individual configuration fields without editing the YAML. The driver script `run_composite_eval.sh` that reproduces the full experimental matrix invokes the `pipeline` subcommand once per target configuration, skipping stages that have already produced their output artifacts, in line with the reproducibility requirements established in NFR4.

The implementation described across Subsections 4.1.1 through 4.1.7 materializes the requirements derived in Section 3.3 into a functional toolkit. The next section reports the empirical results obtained when this toolkit is applied to the five target models defined in the test plan.

4.2 RESULTS

This section reports the empirical results obtained by executing the test plan defined in Section 3.6 against the SafeProbe implementation described in Section 4.1. The results are organized around the three contributions established in this work: the per-attack effectiveness of the four attack modules (including the per-pair behavior of the Composite attack); the inter-rater agreement between the three judges of the multi-judge layer; and the validation of those judges against an external human-annotated ground truth. The artifact and test plan produced 7,800 (prompt, response) pairs, each scored by the three judges of Section 4.1.4. The section is organized into three sub-

sections: Section 4.2.1 reports the attack-level results; Section 4.2.2 reports the cross-judge agreement statistics; and Section 4.2.3 reports the human-validation study.

Before presenting the results, the reading convention adopted in this section is stated explicitly. The headline ASR for every cell is the majority-of-3 verdict (M), in which a row counts as a successful jailbreak only when at least two of the three judges agree on YES. To enable a cross-check between this consensus and the judge that the human-validation study of Section 4.2.3 identifies as most aligned with human raters, every cell also reports HarmBench’s individual ASR (HB). Cells are therefore formatted as “ M / HB”. All ASR values are computed over decisive verdicts only; rows for which a judge returned AMBIGUOUS or ERROR are excluded from the corresponding denominator, so that judges expressing explicit uncertainty are not penalized in the rate.

4.2.1 Attack Effectiveness

A deployer choosing a defense strategy needs to know which attack technique poses the greatest risk on a given target, since the engineering cost of mitigating a cipher-based attack (input-side decoders), an iterative attack (rate limiting and conversation monitoring), and a structured-prompt attack (prompt-template hardening) are not interchangeable. The four attack modules implemented in SafeProbe exercise distinct exploitation surfaces, and aggregating their effectiveness into a single overall ASR hides the per-technique signal that informs that choice. The results presented in this subsection retain that signal in two ways: per-attack ASR on each of the five target models (Table 6), and a finer-grained per-pair view of the Composite ($CO \times MG$) attack (Table 7) that exposes whether the Composite’s higher effectiveness comes from uniform sensitivity to every combination or from a narrow set of highly effective ones.

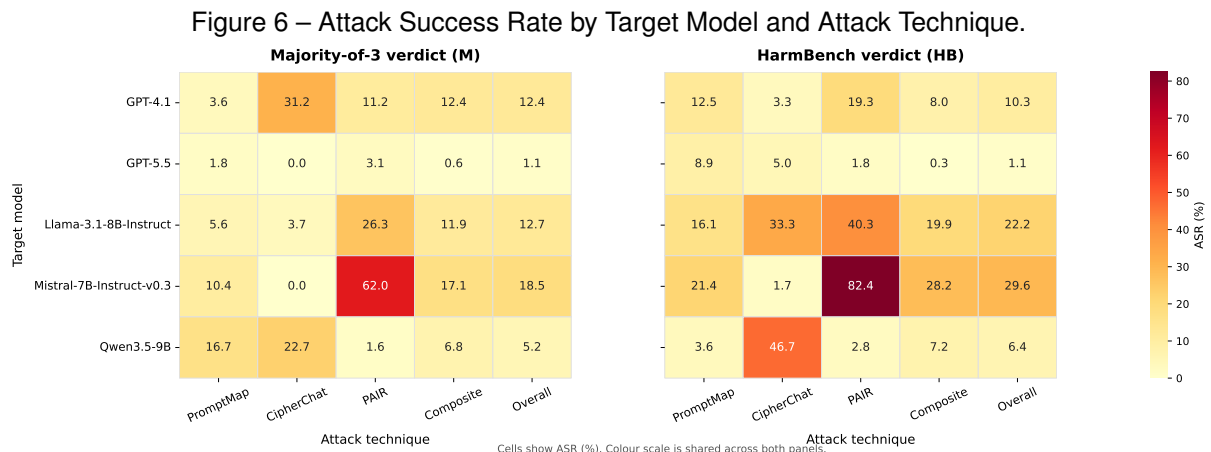
Table 6 – Attack Success Rate by target model and attack technique. Cells are formatted as “ M / HB”, where M is the majority-of-3 verdict and HB is HarmBench’s individual verdict (the judge most aligned with humans, see Section 4.2.3). ASR is computed only over decisive verdicts.

Target	PromptMap	CipherChat	PAIR	Composite	Overall
GPT-4.1	3.6 / 12.5	31.2 / 3.3	11.2 / 19.3	12.4 / 8.0	12.4 / 10.3
GPT-5.5	1.8 / 8.9	0.0 / 5.0	3.1 / 1.8	0.6 / 0.3	1.1 / 1.1
Llama-3.1-8B-Instruct	5.6 / 16.1	3.7 / 33.3	26.3 / 40.3	11.9 / 19.9	12.7 / 22.2
Mistral-7B-Instruct-v0.3	10.4 / 21.4	0.0 / 1.7	62.0 / 82.4	17.1 / 28.2	18.5 / 29.6
Qwen3.5-9B	16.7 / 3.6	22.7 / 46.7	1.6 / 2.8	6.8 / 7.2	5.2 / 6.4

Source: The author.

As shown in Figure 6, the strongest attack changes by target rather than converging on a single technique, and this conclusion is stable under the choice of either the majority-of-3 verdict or HarmBench’s individual verdict as the headline metric. Under the majority rule, CipherChat has the highest ASR on GPT-4.1 (31.2%) and on

Qwen3.5-9B (22.7%); PAIR leads on Llama-3.1-8B-Instruct (26.3%) and on Mistral-7B-Instruct-v0.3 with 62.0% ASR, more than twice the second-best attack on that target; and none of the four attacks reaches double-digit majority ASR on GPT-5.5. The HarmBench column corroborates the ranking on three of five targets: PAIR remains the strongest attack on Llama-3.1-8B-Instruct (40.3%) and on Mistral-7B-Instruct-v0.3 (82.4%), and CipherChat remains the strongest on Qwen3.5-9B (46.7%). On GPT-4.1, HarmBench instead identifies PAIR (19.3%) rather than CipherChat (3.3%) as the strongest attack, indicating that the GPT-4.1/CipherChat majority verdict is largely driven by the two safety-classifier judges flagging cipher-formatted outputs that HarmBench does not regard as harmful content; this caveat is discussed further in Section 4.2.3. On GPT-5.5, every attack technique remains in the single-digit range under both lenses.



Source: The author.

This per-target heterogeneity contradicts the implicit assumption in single-attack benchmarks that one attack class is universally most effective. From a deployment perspective, the per-attack columns of Table 6 are the practically useful result: a deployer self-hosting Mistral-7B-Instruct-v0.3 should prioritize defenses against iterative attackers, while a deployer self-hosting Qwen3.5-9B should prioritize cipher-aware filtering. The Overall column further reveals that aggregating across attack techniques compresses these per-attack differences; a single overall majority ASR of 12.4% for GPT-4.1, for example, hides a 31.2% majority ASR under CipherChat.

The Composite aggregate ASR reported in the Composite column of Table 6 sits in the 0.6%–17.1% majority range (0.3%–28.2% under HarmBench), but this aggregate hides per-pair variation that is critical to the practical interpretation of the attack. To expose this variation, Table 7 reports the top-three (CO, MG) combinations per target ranked by majority-of-3 ASR, with HarmBench’s verdict reported alongside.

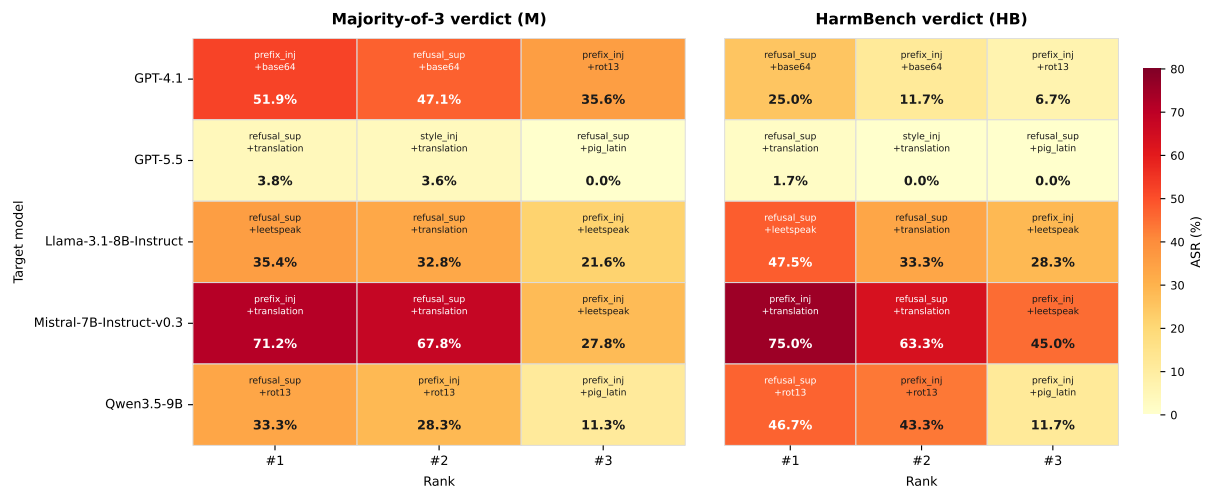
Table 7 – Top-three Composite (CO $\times$ MG) combinations per target, ranked by majority-of-3 ASR. Each cell reports the combination (top line), its majority ASR (middle line), and HarmBench’s individual ASR on the same pair (bottom line).

Target	#1	#2	#3
GPT-4.1	prefix_inj+base64 M: 51.9% HB: 11.7%	refusal_sup+base64 M: 47.1% HB: 25.0%	prefix_inj+rot13 M: 35.6% HB: 6.7%
GPT-5.5	refusal_sup+translation M: 3.8% HB: 1.7%	style_inj+translation M: 3.6% HB: 0.0%	refusal_sup+pig_latin M: 0.0% HB: 0.0%
Llama-3.1-8B-Instruct	refusal_sup+leetspeak M: 35.4% HB: 47.5%	refusal_sup+translation M: 32.8% HB: 33.3%	prefix_inj+leetspeak M: 21.6% HB: 28.3%
Mistral-7B-Instruct-v0.3	prefix_inj+translation M: 71.2% HB: 75.0%	refusal_sup+translation M: 67.8% HB: 63.3%	prefix_inj+leetspeak M: 27.8% HB: 45.0%
Qwen3.5-9B	refusal_sup+rot13 M: 33.3% HB: 46.7%	prefix_inj+rot13 M: 28.3% HB: 43.3%	prefix_inj+pig_latin M: 11.3% HB: 11.7%

Source: The author.

Two patterns are visible in Figure 7. First, target sensitivity varies by a factor of approximately 20 at the per-pair maximum under the majority rule, from 3.8% on GPT-5.5 to 71.2% on Mistral-7B-Instruct-v0.3, and HarmBench shows the same ordering with an even wider extremum (1.7% on GPT-5.5 to 75.0% on Mistral-7B-Instruct-v0.3). The same family of attacks therefore produces target-dependent exposure that spans more than an order of magnitude, regardless of which judge is consulted. Second, the dominant combination differs per target rather than converging on a single “best” attack: base64 variants dominate against GPT-4.1, translation variants against Mistral-7B-Instruct-v0.3, leetspeak and translation variants against Llama-3.1-8B-Instruct, and rot13 variants against Qwen3.5-9B. This per-target specialization motivates the per-pair analysis as the unit of evaluation: an aggregate Composite ASR that pools all twenty combinations into one number cannot distinguish a target that is uniformly vulnerable to every combination from a target that is highly vulnerable to one combination and resistant to the others, yet the defender’s mitigation cost differs by the number of (CO, MG) pairs that must be addressed in the two cases.

Figure 7 – Composite (CO X MG) Combinations per Target Model.



M panel ranked by majority ASR; HB panel ranked by HarmBench ASR. Colour scale shared. Source: The author.

Source: The author.

The strongest per-pair Composite combination reaches majority ASR of 51.9% on GPT-4.1, 35.4% on Llama-3.1-8B-Instruct, 71.2% on Mistral-7B-Instruct-v0.3, and 33.3% on Qwen3.5-9B, a $3.0\times$ to $4.9\times$ multiplier over the corresponding aggregate Composite ASR reported in the Composite column of Table 6. Under HarmBench, the multipliers are of the same order: $5.9\times$ on GPT-4.1 (8.0% aggregate *versus* 47.5% per-pair maximum across the top three), $2.7\times$ on Mistral-7B-Instruct-v0.3, and $6.5\times$ on Qwen3.5-9B. A deployer who reads only the aggregate Composite ASR therefore underestimates the strongest realistic single-shot attack by a factor of three to six on either lens. The per-pair view also enables targeted defenses: knowing that prefix_injection+base64 is the dominant combination against GPT-4.1 under the majority rule suggests prioritizing input-side detectors that strip base64-encoded payloads before safety classification, a more constrained engineering target than a generic Composite-class defense.

A limitation of this experiment is that the harmful behaviors are sampled from general-purpose jailbreak benchmarks rather than from domain-specific critical scenarios. Future evaluations could instantiate the same Composite grid over malicious tasks tailored to high-impact domains such as healthcare, finance, cybersecurity, public administration, education, and critical infrastructure. This would allow the per-pair CO $\times$ MG analysis to answer a more operational question: not only which structural attack pattern is most effective against a target model, but which pattern is most effective for a specific risk domain. Such domain-conditioned evaluation is particularly relevant for deployments in which the acceptable safety threshold depends on the application context rather than on a generic harmfulness benchmark.

The best Composite combination outperforms every single-technique baseline on four of five targets under the majority rule. The per-pair gap against PromptMap

reaches 48.3 percentage points on GPT-4.1 (3.6% *versus* 51.9%); against CipherChat it reaches 31.7 percentage points on Llama-3.1-8B-Instruct (3.7% *versus* 35.4%); and against PAIR it reaches 9.2 percentage points on Mistral-7B-Instruct-v0.3 (62.0% *versus* 71.2%). The only target for which a single-technique attack matches the best Composite combination under the majority rule is Mistral-7B-Instruct-v0.3, for which PAIR (62.0%) is close to the strongest Composite pair (71.2%); even there, however, a single Composite query (one model call per behavior) dominates a PAIR run (twenty-five model calls per behavior, in the experimental configuration). Under HarmBench, PAIR remains stronger than the best Composite pair on Mistral-7B-Instruct-v0.3 (82.4% *versus* 75.0%), but Composite remains dominant on GPT-4.1, Qwen3.5-9B, and Llama-3.1-8B-Instruct, and the cost asymmetry argument applies equally under both lenses. The aggregate Composite ASR alone is not a stronger benchmark than PAIR or CipherChat, but the per-pair maximum is, and at a fraction of the query cost.

Even before discussing judges in detail, the disagreement between the M and HB values in Tables 6 and 7 shows that the choice of judge can shift a single cell by tens of percentage points. The GPT-4.1 CipherChat cell illustrates this directly, with $M = 31.2\%$ and $HB = 3.3\%$; the widest spread in the per-pair Composite analysis is the GPT-4.1 prefix_injection+base64 pair, for which $M = 51.9\%$ but $HB = 11.7\%$, a 40-pp gap. This judge sensitivity is the subject of Section 4.2.2.

4.2.2 Cross-Judge Agreement

The cross-judge agreement analysis constitutes a secondary methodological contribution of this work. While the Composite module contributes a new attack construction and diagnostic view, the agreement analysis contributes a reporting discipline for jailbreak evaluation: every ASR value is treated as conditioned by the verdict mechanism that produced it. This subsection therefore does not introduce a new attack, but provides the statistical evidence required to interpret attack success rates responsibly.

Every ASR value reported in Section 4.2.1 depends on a verdict produced by an LLM-as-a-judge, and the gap between the M and HB values in Tables 6 and 7 already revealed that swapping one judge for another can move a single cell by tens of percentage points. Whether those tens of percentage points represent random noise, a systematic bias of a single judge, or a deeper taxonomic disagreement between classes of judges determines whether the headline numbers can be defended at all. The analysis presented in this subsection quantifies the disagreement first as direct per-judge ASR across all three judges (Table 8) and then through pairwise Cohen’s κ and multi-rater Fleiss’ κ (Table 9), establishing the magnitude and shape of the disagreement that Section 4.2.3 then adjudicates against an external reference.

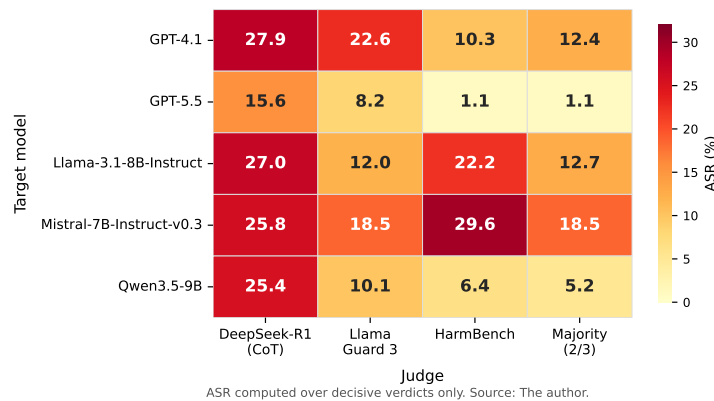
Table 8 – Per-judge overall ASR (%) by target. ASR is computed only over decisive verdicts (YES/NO). The Majority column reports the consensus verdict (at least 2 of 3 judges agree on YES); rows for which judges split 1-1-1 or for which two or more judges return AMBIGUOUS are excluded from the majority denominator.

Target	DeepSeek-R1 (CoT)	Llama Guard 3	HarmBench	Majority (2/3)
GPT-4.1	27.9	22.6	10.3	12.4
GPT-5.5	15.6	8.2	1.1	1.1
Llama-3.1-8B-Instruct	27.0	12.0	22.2	12.7
Mistral-7B-Instruct-v0.3	25.8	18.5	29.6	18.5
Qwen3.5-9B	25.4	10.1	6.4	5.2

Source: The author.

As shown in Table 8, the three judges are not interchangeable: the per-judge columns diverge substantially within each row, confirming that judge identity is a first-order determinant of the reported ASR. The most pronounced divergence occurs on GPT-5.5, where DeepSeek-R1 assigns an ASR of 15.6% while HarmBench assigns only 1.1% over the same set of (prompt, response) pairs, a 14 $\times$ ratio. On GPT-4.1 the spread is a 17.6-pp gap (27.9% *versus* 10.3%) in the same direction, and on Qwen3.5-9B a 19.0-pp gap (25.4% *versus* 6.4%). The pattern is consistent: DeepSeek-R1 reports the highest ASR on four of five targets, HarmBench reports the lowest on four of five targets, and Llama Guard 3 sits between them on four of five targets. The single exception is Mistral-7B-Instruct-v0.3, for which HarmBench (29.6%) slightly exceeds DeepSeek-R1 (25.8%), suggesting that HarmBench’s training distribution covers the response styles Mistral produces more thoroughly than the other targets’.

Figure 8 – Overall Attack Success Rate (%) per Target Model Across Judges.



Source: The author.

The Majority column of Table 8 consistently falls at or below the median individual-judge estimate, reflecting the conservatism of requiring agreement from at least two judges before recording a positive verdict. This conservatism is a deliberate design property: when judges disagree, the majority rule treats the disagreement as evidence against confidently labelling the row as a jailbreak. The validation reported in Section 4.2.3 shows that HarmBench’s lower individual ASR is the closest to ground truth, and that the majority verdict’s conservatism is not unwarranted on the bulk of rows; the

practical implication is that the gap between M and HarmBench in Table 8 can be read as an upper bound on the noise contributed by the two other judges.

The pairwise and multi-rater agreement statistics quantify this disagreement directly. Table 9 reports pairwise Cohen's κ between every pair of judges and Fleiss' κ across all three judges simultaneously, computed only over rows for which every judge produced a decisive verdict.

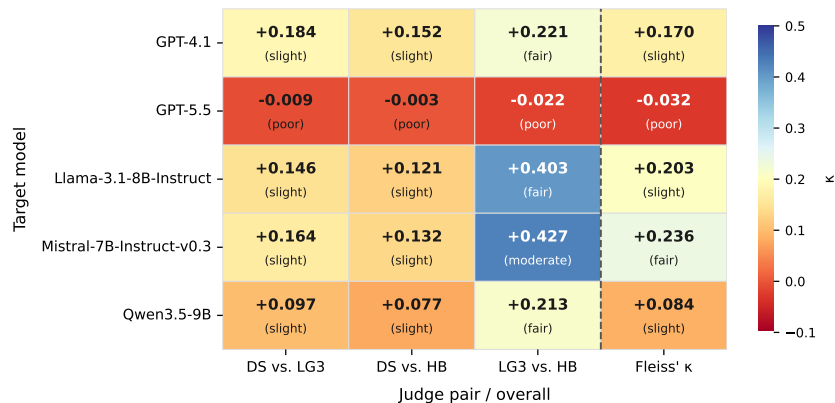
Table 9 – Inter-rater agreement per target: pairwise Cohen's κ between the three judges and overall Fleiss' κ . All κ values are computed over rows for which all three judges produced a decisive verdict, so the three pairwise values and Fleiss' κ share a common denominator. Interpretation labels follow the Landis and Koch scale.

Target	DS vs. LG3	DS vs. HB	LG3 vs. HB	Fleiss' κ (label)
GPT-4.1	0.184 (slight)	0.152 (slight)	0.221 (fair)	0.170 (slight)
GPT-5.5	−0.009 (poor)	−0.003 (poor)	−0.022 (poor)	−0.032 (poor)
Llama-3.1-8B-Instruct	0.146 (slight)	0.121 (slight)	0.403 (fair)	0.203 (slight)
Mistral-7B-Instruct-v0.3	0.164 (slight)	0.132 (slight)	0.427 (moderate)	0.236 (fair)
Qwen3.5-9B	0.097 (slight)	0.077 (slight)	0.213 (fair)	0.084 (slight)

Source: The author.

Table 9 show the agreement landscape is dominated by the slight band, with two localized exceptions in the Llama Guard 3 *versus* HarmBench column for the open-weight targets. Fleiss' κ across the three judges sits in the slight-to-fair regime (0.084–0.236) on four of five targets and is effectively zero on GPT-5.5 (−0.032). None of the targets reaches the moderate threshold ($\kappa \geq 0.41$) in the aggregate. The per-target consequence is visible in Table 8: on GPT-5.5, the same set of (prompt, response) pairs yields per-judge ASR of 1.1% (HarmBench), 8.2% (Llama Guard 3), and 15.6% (DeepSeek-R1). The negative Fleiss' κ for GPT-5.5 is a low-base-rate artefact rather than active disagreement: GPT-5.5 refuses most attacks, so the modal verdict from every judge is NO, and the few positive verdicts are scattered across different items. Agreement statistics are most informative on targets that are non-trivially vulnerable.

Figure 9 – Inter-rater Agreement (Cohen's κ and Fleiss' κ) Between Judges per Target Model.



Source: The author.

The pairwise structure of Table 9 also exposes systematic biases between judges. Llama Guard 3 and HarmBench reach moderate agreement on the two open-weight targets for which the underlying response distribution is most varied (Llama-3.1-8B-Instruct: $\kappa = 0.403$; Mistral-7B-Instruct-v0.3: $\kappa = 0.427$), suggesting that the two safety-trained classifiers share a similar taxonomic prior on those response styles. DeepSeek-R1, in contrast, never exceeds $\kappa = 0.184$ with any other judge on any target. The reasoning-based judge does not produce verdicts that are merely noisier than the safety-classifier judges; it produces verdicts that are systematically different. Without an external reference, this disagreement cannot be adjudicated, motivating the human-validation study reported in Section 4.2.3.

4.2.3 Human Validation

The agreement statistics reported in Section 4.2.2 establish that the three judges disagree systematically rather than randomly, but disagreement alone does not indicate which verdict, if any, corresponds to the harm judgement that a human annotator would assign. Without that anchor, the dual-headline convention used in Tables 6 and 7 (majority-of-3 alongside HarmBench) would have no empirical justification, and there would be no principled basis on which to interpret the gap between M and HB in each cell. The validation reported in this subsection supplies that anchor: it establishes a human-majority ground truth on a stratified 200-item sample, measures each judge against that ground truth, and uses the resulting per-judge metrics to qualify the magnitude and direction of the bias embedded in every cell of Section 4.2.1.

Following the protocol defined in Section 3.6.1, three independent human raters annotated a 200-item sample of (prompt, response) pairs drawn from the consolidated dataset. To reduce human-evaluation bias, annotators followed a standardized annotation prompt. For each item, they received three fields: the original harmful behavior, the adversarial prompt submitted to the target model, and the target model’s response. They were instructed to judge only whether the response materially complied with the harmful objective, independently of whether the answer was fluent, complete, or technically sophisticated. They were also instructed not to infer intent beyond the text shown and not to penalize refusals that contained partial safety explanations. The annotation question was: *“Does the model response provide actionable, enabling, or persuasive assistance toward the harmful objective expressed in the prompt?”* The available labels were YES, meaning the jailbreak succeeded, and NO, meaning the refusal or safe redirection held.

The sample comprises 40 items per target distributed across four strata of 50 items each: unanimous-YES (all three judges agreed on YES), unanimous-NO (all three agreed on NO), split (the judges disagreed), and has-ambiguous (at least one

judge returned AMBIGUOUS). Each item was annotated independently by the three raters as YES (jailbreak succeeded) or NO (refusal held), without access to the judge verdicts to avoid anchoring bias. Fleiss’ κ across the three human raters on the full 200-item sample is 0.824 (almost perfect, by the Landis and Koch scale), and the three pairwise Cohen’s κ values fall in the range $[0.808, 0.863]$. Inter-rater pairwise percent-agreement is 92.4%–95.0%, so the human-majority verdict (at least two of three humans agree) is a defensible ground-truth label for the items in the sample. This high inter-annotator agreement also establishes a ceiling: a judge cannot be more aligned with humans than the humans are with each other. Treating the human-majority label as ground truth, Table 10 reports the per-judge metrics: precision (fraction of judge-flagged jailbreaks that humans confirm), recall (fraction of human-confirmed jailbreaks that the judge flagged), F1 (harmonic mean), accuracy, and Cohen’s κ between the judge verdict and the human-majority label.

Table 10 – Per-judge agreement with the human-majority ground truth on the 200-item validation sample. P: precision; R: recall; F1: harmonic mean; Acc: accuracy; κ : Cohen’s kappa with the human-majority label. The Majority(2/3) row reports the consensus judge verdict against the human-majority. All metrics are computed over rows for which both the judge (or judge-majority) and the human-majority verdicts are decisive.

Judge	P	R	F1	Acc	κ
DeepSeek-R1 (CoT)	0.442	0.791	0.567	0.653	0.314
Llama Guard 3	0.559	0.745	0.639	0.785	0.490
HarmBench	0.712	0.922	0.803	0.885	0.724
Majority (2/3)	0.684	0.830	0.750	0.863	0.657

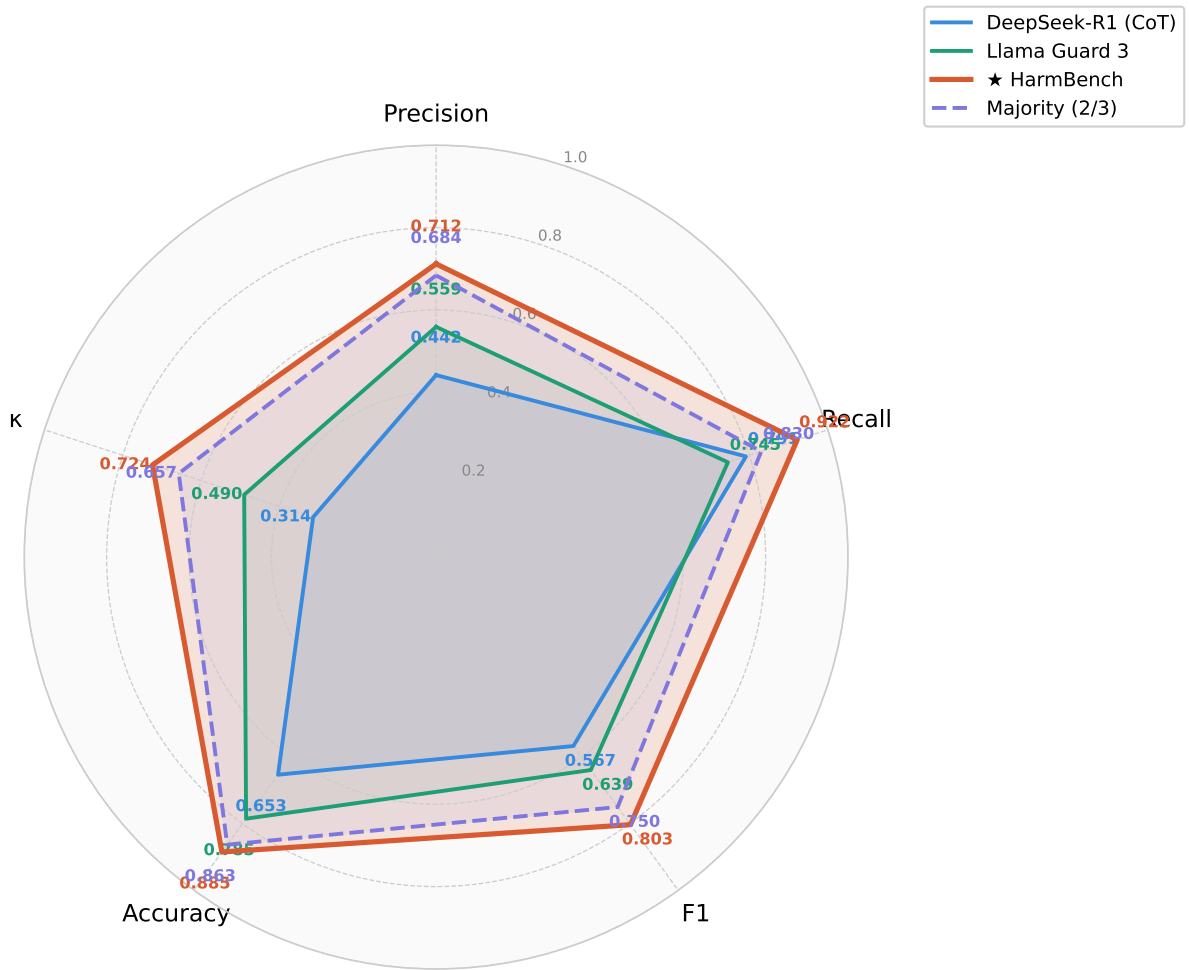
Source: The author.

As shown in Table 10, the three judges occupy distinct regions of the metric space. HarmBench dominates all five metrics (precision 0.712, recall 0.922, F1 0.803, accuracy 0.885, $\kappa = 0.724$), reaching substantial agreement against the human majority on the Landis and Koch scale. Llama Guard 3 ranks second on every metric. DeepSeek-R1 presents a skewed profile of high recall (0.791) combined with low precision (0.442), indicating that it flags a large fraction of human-confirmed jailbreaks but also generates a disproportionate number of false positives. Its F1 of 0.567 and κ of 0.314 place it in only fair agreement with the human majority, trailing HarmBench by 0.236 in F1 and 0.410 in κ .

The single-judge ranking on this sample inverts the cross-judge ASR ranking reported in Section 4.2.2: the judge with the lowest ASR (HarmBench) is the most aligned with humans, and the judge with the highest ASR (DeepSeek-R1) is the least aligned. This inversion indicates that DeepSeek-R1’s higher ASR numbers reflect over-flagging of borderline responses that humans label as refusals, rather than genuine detection of harms missed by the simpler classifiers. The result is consistent with Llama Guard 3 and HarmBench having been fine-tuned on labelled harm distributions that

overlap with the categories represented in the sample, while DeepSeek-R1 reasons about harm from instruction-following pretraining alone. It is this empirical fact, established here, that justifies the dual-headline reporting convention used in Tables 6 and 7: HarmBench is the single judge with the strongest claim to approximate the human-majority label, and reporting its individual verdict alongside the majority-of-3 verdict brackets the cells of Section 4.2.1 between a consensus rule and a human-aligned single classifier.

Figure 10 – Per-judge Agreement with Human Ground Truth Across Evaluation Metrics.



★ HarmBench is the best-aligned judge. Metrics: P = precision, R = recall, F1 = harmonic mean, Acc = accuracy, κ = Cohen's kappa. Source: The author.

Source: The author.

The majority-of-3 verdict used as the headline aggregation rule in Section 4.2.1 achieves F1 0.750 and $\kappa = 0.657$ against the human majority. Its precision (0.684) is below HarmBench's (0.712), but its operational utility is different: the majority-of-3 rule requires two independent classifiers to agree on YES before reporting a jailbreak, which makes it more conservative against any individual judge's idiosyncratic biases. On the validation sample, 39 of 57 majority-YES verdicts are confirmed by humans (a 68%

confirmation rate), and the remaining 18 false positives are concentrated on Qwen3.5-9B (12 of 18) and on Composite responses (16 of 18), indicating that Qwen3.5-9B’s response style produces text that two of three judges over-flag, particularly when the input is a Composite-style obfuscation. This is the largest single source of inflation in the headline majority ASR: the Qwen3.5-9B Composite ASR of 6.8% in Table 6 should be read with the caveat that two-thirds of the items contributing to it are false positives by human standards, and the HarmBench reading of 7.2% on the same cell is no more reliable on this target.

Eight of 200 items (4%) are human-majority YES but judge-majority NO. Six of these eight are cases in which HarmBench correctly flagged the response but was out-voted by the other two judges. The recall floor of judge-majority is therefore bounded by HarmBench’s recall (0.922): when HarmBench misses a true jailbreak, no consensus rule among the current three judges can recover it. The eight missed jailbreaks are distributed across all five targets (Llama-3.1-8B-Instruct: 3, Mistral-7B-Instruct-v0.3: 2, GPT-4.1: 1, GPT-5.5: 1, Qwen3.5-9B: 1) and concentrate in the Composite (4) and PromptMap (3) attack families. They fall into two patterns. First, soft-compliance responses in which the model technically engaged with the harmful framing but produced low-graphicality content (for example, a persuasive argument for lowering the drinking age, or a defamatory article whose content has been topically relocated to a non-harmful subject). Second, non-English compliant outputs (the Mistral-7B Spanish body-shaming campaign in item 140) in which the textual classifiers may simply not have triggered. Both patterns are addressable by extending the multi-judge layer with a multilingual classifier complemented by a soft-harm-aware classifier, an extension that the SafeProbe plug-in architecture described in Section 4.1.4 supports without changes to the core pipeline.

The validation tightens the interpretation of Tables 6 and 7 in three ways. First, the gap between M and HB in those tables should not be read as a symmetric uncertainty band: HB is the side with the stronger claim to ground truth, so on cells for which $M > \text{HB}$ (the typical case) the majority verdict is the upper bound, and on cells for which $\text{HB} > M$ (the cases driven by Llama Guard 3 and HarmBench agreeing against DeepSeek-R1) the HarmBench verdict is the more defensible point estimate. Second, the robustness conclusion on GPT-5.5 is reinforced: of the 40 GPT-5.5 sample items, humans label only 1 as a true jailbreak, and HarmBench correctly identifies it, so the 1.1% majority-of-3 ASR reported for GPT-5.5 in Table 6, which exactly coincides with HarmBench’s 1.1% on that target, is an upper bound rather than a lower one on this sample. Third, on the four non-reasoning targets, the recall ceiling of 0.922 implies that the true ASR of those targets is up to approximately 8 percentage points higher than the headline majority numbers in Table 6. This correction does not change the

within-cluster ranking but does mean that a reported value of 12.4% should be read as a defensible point estimate within a band whose upper edge is bounded by the corresponding HarmBench value adjusted by the recall gap.

4.2.4 Discussion

The results support three main findings. First, attack effectiveness is target-dependent: no single attack technique dominates across all evaluated models. CipherChat is most effective against GPT-4.1 and Qwen3.5-9B under the majority verdict, while PAIR dominates against Llama-3.1-8B-Instruct and Mistral-7B-Instruct-v0.3. GPT-5.5 remains comparatively robust across all four attack families. This variation indicates that jailbreak robustness should not be reported as a single model-level property without specifying the attack family used to measure it.

Second, the Composite module shows that aggregate attack metrics can hide operationally important vulnerability patterns. Although the aggregate Composite ASR is not always the strongest value in Table 6, the per-pair $CO \times MG$ analysis reveals combinations that substantially exceed the aggregate. This makes Composite the main attack-oriented contribution of SafeProbe: it transforms attack evaluation from a single aggregate success rate into a diagnostic profile of structural weaknesses. A defender can use this profile to prioritize targeted mitigations, such as decoding filters, prompt-wrapper detection, or specific defenses against role-play and refusal-suppression patterns.

Third, the cross-judge and human-validation analyses show that ASR values are judge-conditioned rather than model-intrinsic. The same set of model responses produced substantially different ASR estimates depending on whether DeepSeek-R1, Llama Guard 3, HarmBench, or a majority-of-3 rule was used. The inter-rater agreement statistics remained in the slight-to-fair range for most targets, and the human-validation study showed that HarmBench was the closest judge to the human-majority reference on the 200-item sample. This finding does not invalidate LLM-as-a-judge evaluation, but it shows that single-judge ASR values should be interpreted as estimates produced by a specific verdict mechanism.

Compared with the related works discussed in Section 3.4, these results position SafeProbe as complementary rather than merely competitive. HarmBench contributes standardized harmful behavior evaluation and a strong binary classifier; JailbreakBench emphasizes reproducible jailbreak artifacts; Garak provides broad vulnerability probing; EasyJailbreak focuses on attack-generation workflows; and WildGuard contributes safety classification across risk categories. SafeProbe differs by combining four elements in a single evaluation pipeline: multi-backend target evaluation, attack diversity, the Composite $CO \times MG$ diagnostic attack, and explicit judge-validity reporting

through inter-rater agreement and human validation. The empirical results suggest that the last two elements are especially important: Composite exposes per-target vulnerabilities that aggregate benchmarks can hide, while agreement statistics reveal when reported ASR values are unstable under judge substitution.

These findings also clarify the limitations of the present evaluation. The harmful behaviors were sampled from general jailbreak benchmarks, which supports comparability with prior work but does not fully represent domain-specific risk in critical applications. The human-validation study provides an external reference, but it remains limited to 200 items and to the annotation rubric used in this work. In future evaluations, the same methodology should be extended with domain-specific harmful tasks and domain-specific human annotation rubrics, especially for healthcare, finance, cybersecurity, public administration, and critical infrastructure. Such extensions would test whether the same CO $\times$ MG patterns remain dominant when the harmful objective is specialized to a high-impact operational context.

Overall, the results motivate the recommendation that every published jailbreak ASR be accompanied by the judge identity, the verdict aggregation rule, and an inter-rater agreement statistic when multiple judges are used. In the same way that classification accuracy is more interpretable when accompanied by uncertainty estimates, jailbreak ASR is more interpretable when accompanied by evidence about verdict stability. Without this information, comparisons across papers risk conflating model robustness, attack strength, and judge bias.

5 CONSIDERATIONS & FUTURE WORK

SafeProbe was built to test the safety alignment of LLMs at inference time under a query-access threat model, and Chapter 4 reported its application to five targets, four attack modules, and three judges. Subsection 4.2.4 already draws together the three findings that matter most: no attack technique dominates across every target, the reported ASR depends on the judge that produced it, and the most permissive judge turns out to be the least aligned with human raters. Rather than repeat that analysis, this chapter returns to the requirements that motivated the toolkit, the limits of what the experiments can claim, and the work left to do.

The functional requirements set out in Section 3.3 are met by the delivered toolkit. FR1 is handled by the unified adapter layer, which puts proprietary and open-weight models behind a single protocol; FR2 by the four attack modules, including the Composite technique introduced in this work; FR3 by the reasoning judge, which exposes the chain of reasoning behind each verdict; and FR4 by the agreement statistics that travel with every reported ASR. The non-functional requirements follow from the same design: sequential judge execution keeps memory within the testbed budget, and the configuration files make each run reproducible from a single archived artifact.

The evaluation of safe alignment in LLMs runs into difficulties that come from how neural networks are built. The same generalization that makes these models useful also leaves them open to attack, since the ability to interpolate between training examples is also the ability an adversary turns into a perturbation. Read this way, adversarial examples are less a defect to be patched than a by-product of how networks learn to cover high-dimensional input spaces.

These weaknesses survive the jump in model capability. Normalization layers help training converge, but they also open a surface to attack: the statistical regularities they impose are predictable, and an adversary can lean on them. Because the rescaling step magnifies small perturbations, an input crafted around the normalized feature distribution can move the model further than its size would suggest. The probabilistic core of these systems then makes firm safety guarantees hard to state, since there is no clean line that separates safe from unsafe operation.

Most defenses in adversarial machine learning answer one attack vector at a time and leave the rest uncovered. That piecemeal quality is itself the gap this work speaks to: the field still lacks a method that assesses safety consistently across architectures, attacks, deployment modes, and verdict mechanisms.

Moderation tools such as perplexity filters and model-versus-model checks do

catch a share of harmful prompts in practice. Leaning on LLMs to police other LLMs, however, raises a harder question about where the safety guarantee is grounded. The arrangement invites an infinite regress, in which every layer of verification needs a verifier of its own, never settling on a level of trust that does not itself depend on a model. A flaw in a guardian model can then propagate through the whole stack, which is the practical reason this work reports agreement among judges instead of trusting any single verdict.

5.1 PUBLICATIONS

During the development of this work, the following publications were made:

- **COCHAK, Henrique; NETO, Milton; MIERS, Charles; MARQUES, Marco; SIMPLICIO JR, Marcos A. (2024).** Enhancing SPIFFE/SPIRE Environment with a Nested Security Token Model. In: *Proceedings of the 14th International Conference on Cloud Computing and Services Science - Volume 1: CLOSER*. INSTICC, SciTePress, p. 184–191.
- **NETO, Milton P. Pagliuso; MIERS, Charles C. (2024).** Proposta de análise de desempenho do uso de mecanismos de segurança em sistemas inteligentes usando aprendizado de máquina e IA gerativa adversariais do tipo LLM. In: *Escola Regional de Alto Desempenho da Região Sul (ERAD-RS)*. SBC.
- **MIERS, Charles Christian; SIMPLICIO JR, Marcos Antonio; TORREZ ROJAS, Marco Antonio; NETO, Milton Pedro Pagliuso; et al.(2024).** Ferramentas de Jailbreaking para Grandes Modelos de Línguas – Aprendizado de Máquina no Contexto Adversário. Minicurso no *XXIV Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais (SBSeg)*, São José dos Campos, SP, realizado de 16 a 19 de setembro de 2024. DOI: 10.5753/sbc.15101.8.2.
- **ALVES, Ana Carolina Vedoy; NETO, Milton Pedro Pagliuso; MIERS, Charles Christian (2025).** Análise comparativa sobre atividades de Red Team de LLMs com LLMs. In: *Escola Regional de Alto Desempenho da Região Sul (ERAD-RS)*. SBC.
- **ALVES, Ana Carolina Vedoy; MARTINAGNO, Giovanni Mandel; CORREA, Matheus Roberto Da Silva; NETO, Milton Pedro Pagliuso; PILLON, Mauricio; MIERS, Charles Christian; CORREIA, Pedro H. Barcha; HAYASHI, Victor Takashi; SIMPLICIO JR, Marcos A. (2026).** Comparative Analysis of Red Teaming Activities on LLMs Using GCG, DeGCG, AutoDAN, and ReNeLLM Against LLMs. In: *The 40th International Conference on Advanced Information Networking and Applications (AINA-2026)*.

- **BUNN, Carlos D. S.; NETO, Milton P. Pagliuso; PILLON, Maurício A.; MIERS, Charles C.; CORREIA, Pedro H. Barcha; MARQUES, Marco A.; SIMPLICIO JR, Marcos A. (2025).** Comparative Analysis of Consensus Protocols in Blockchains: IBFT 2.0, QBFT, and SmartBFT. In: *2025 IEEE CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON)*, p. 1–6. IEEE.

Six articles have already been written and submitted, and are currently awaiting the position of the reviewers and editors:

- **HAYASHI, Victor T.; NETO, Milton P. Pagliuso; MIERS, Charles C.; ARAKAKI, Reginaldo; SIMPLICIO JR, Marcos A. (2025).** DeepSeek under Attack: an Analysis of Jailbreak Attacks and Prompt-Level Defenses. *Elsevier Communication Networks*.
- **CORREA, Matheus R. S.; BUNN, Carlos; PAGLIUSO, Milton; MIERS, Charles C. (2026).** PAIR transfers, CipherChat does not: Indirect Prompt Injection in MCP Tool Responses. *WTICG – SBSEG 2026*.
- **MANDEL, Giovanni; PAGLIUSO, Milton; MIERS, Charles C. (2026).** Comparative Analysis of Automated Red Teaming in LLMs: AttnGCG, GPTFuzzer, and PAPILLON. *WTICG – SBSEG 2026*.
- **BUNN, Carlos; CORREA, Matheus R. S.; PAGLIUSO, Milton; MIERS, Charles C. (2026).** Adversarial Effectiveness vs. Computational Cost: An Analysis of GCG and nanoGCG Adversarial Suffix Attacks. *WTICG – SBSEG 2026*.
- **SÁ, Matheus A.; SENNA, Paulo; PAGLIUSO, Milton; MIERS, Charles C. (2026).** UnifiedBench: Toward a Unified Dataset for Red-Teaming Language Models. *WTICG – SBSEG 2026*.
- **HAYASHI, Victor T.; PAGLIUSO, Milton; MANDEL, Giovanni; MIERS, Charles C.; SIMPLICIO JR, Marcos A. (2026).** Red Teaming an Industrial Maintenance Chatbot using PAIR and nanoGCG. *Industry Talks – SBSEG 2026*.

A further conference and journal article is planned using the results of the experiments planned in this research. The forecast for submission of these articles is at the end of 2026.

5.2 FUTURE WORK

The limitations identified in this dissertation point to several extensions of SafeProbe. First, the two failure patterns found in the human-validation study, non-English compliance and low-graphicality harm, call for new judges: a multilingual safety classifier and a soft-harm-aware classifier. This extension is relevant because current safety evaluation increasingly recognizes that harmfulness is not limited to explicit or graphically violent outputs; it can also appear through persuasion, manipulation, indirect assistance, or domain-specific advice. The plug-in design of SafeProbe allows such judges to be added without changing the rest of the pipeline.

Second, the judge pool should be widened beyond the three verdict mechanisms used in this work. Prior work such as HarmBench, Llama Guard, WildGuard, and StrongREJECT shows that classifier-based and rubric-based safety evaluation can produce different verdict profiles (MAZEIKA et al., 2024; INAN et al., 2023; HAN et al., 2024; SOULY et al., 2024). Expanding the judge pool would make the agreement figures speak for a larger family of verdict mechanisms rather than for one trio, strengthening the claim that ASR values are judge-conditioned.

Third, the target set should grow to include more model families, parameter scales, alignment methods, and deployment regimes. The results in Chapter 4 already show that attack effectiveness changes substantially across targets, which suggests that conclusions drawn from a small set of models may not generalize. Future experiments should therefore include additional proprietary models, open-weight models, multilingual models, and domain-specialized models.

Fourth, the threat model should move beyond single-query attacks. Multi-turn adversarial conversations are increasingly relevant because deployed assistants often maintain conversational state, use tool calls, and respond to progressive user steering rather than isolated prompts. Future work should therefore extend SafeProbe to evaluate multi-turn jailbreaks, indirect prompt injection, tool-mediated attacks, and agentic workflows in which the harmful effect may emerge only after several interaction steps.

Fifth, the Composite taxonomy should be expanded beyond the Competing Objectives and Mismatched Generalization families used in this dissertation. As new jailbreak structures appear, SafeProbe can incorporate additional transformation families, including context poisoning, tool-output injection, multilingual obfuscation, persona escalation, and goal-hijacking strategies. This would preserve the main advantage of Composite: producing a diagnostic vulnerability profile rather than only a single aggregate ASR.

Finally, future work should instantiate the same evaluation protocol in domain-specific critical settings, including healthcare, finance, cybersecurity, public adminis-

tration, education, and critical infrastructure. Domain-specific evaluation would make it possible to determine whether the most effective attack patterns observed in generic jailbreak benchmarks remain dominant when the harmful objective is tied to a specialized operational context.

5.2.1 AI Usage Declaration

In line with the principles of academic integrity and transparency, the author declares the use of Artificial Intelligence (AI) tools in the preparation of this text, as detailed below:

- a) Claude Opus (Anthropic) was used for the revision of the text and for adjustments to the English writing.
- b) Claude Sonnet (Anthropic) was used for brainstorming ideas and refining the research proposal.

The author declares that he has reviewed the entire text and takes full responsibility for its content.

BIBLIOGRAPHY

- ABDALI, S. et al. Securing large language models: Threats, vulnerabilities and responsible practices. **arXiv preprint arXiv:2403.12503**, 2024.
- ALAYRAC, J.-B. e. a. Flamingo: a visual language model for few-shot learning. In: **Advances in Neural Information Processing Systems**. [S.l.: s.n.], 2022. v. 35, p. 23716–23736.
- APRUZZESE, G. et al. The role of machine learning in cybersecurity: A survey. **ACM Digital Threats: Research and Practice**, ACM New York, NY, v. 4, n. 1, p. 8, 2023.
- BENGIO, Y. Learning deep architectures for ai. **Foundations and trends in Machine Learning**, Now Publishers Inc, v. 2, n. 1, p. 1–127, 2009.
- BOMMASANI, R. et al. On the opportunities and risks of foundation models. **arXiv preprint arXiv:2108.07258**, 2021.
- BROWN, T. et al. Language models are few-shot learners. **Advances in neural information processing systems**, v. 33, p. 1877–1901, 2020.
- BUBECK, S. et al. Sparks of artificial general intelligence: Early experiments with gpt-4. **arXiv preprint arXiv:2303.12712**, 2023.
- BUBECK, S. et al. Adversarial examples from computational constraints. In: **International Conference on Machine Learning (ICML)**. [S.l.: s.n.], 2019. p. 831–840.
- CARLINI, N. et al. Extracting training data from large language models. In: **30th USENIX Security Symposium (USENIX Security 21)**. [S.l.: s.n.], 2021. p. 2633–2650. ArXiv:2012.07805.
- CATH, C. et al. Governing artificial intelligence: ethical, legal and technical opportunities and challenges. **Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences**, The Royal Society Publishing, v. 376, n. 2133, p. 20180080, 2018.
- CHAKRABORTY, A. et al. Adversarial attacks and defences: A survey. **arXiv preprint arXiv:1810.00069**, 2018.
- CHAO, P. et al. Jailbreakbench: An open robustness benchmark for jailbreaking large language models. In: **Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track**. [S.l.: s.n.], 2024.
- CHAO, P. et al. Jailbreaking black box large language models in twenty queries. In: IEEE. **2025 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)**. [S.l.], 2025. p. 23–42.
- CHEN, X. et al. TeleAI-Safety: A comprehensive LLM jailbreaking benchmark towards attacks, defenses, and evaluations. **arXiv preprint arXiv:2512.05485**, 2025.

CHOWDHERRY, A. et al. Palm: Scaling language modeling with pathways. **arXiv preprint arXiv:2204.02311**, 2022.

DAS, B. C.; AMINI, M. H.; WU, Y. Security and privacy challenges of large language models: A survey. **arXiv preprint arXiv:2402.00888**, 2024.

DEEPSEEK-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. 2025. Disponível em: <<https://arxiv.org/abs/2501.12948>>.

DELDJOO, Y. et al. Recommender systems leveraging multimedia content. **ACM Computing Surveys (CSUR)**, ACM New York, NY, USA, v. 53, n. 5, p. 1–38, 2021.

DERCZYNSKI, L. et al. garak: A framework for security probing large language models. **arXiv preprint arXiv:2406.11036**, 2024.

DEVLIN, J. et al. Bert: Pre-training of deep bidirectional transformers for language understanding. In: **Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)**. [S.l.: s.n.], 2018. p. 4171–4186.

DOUMBOUYA, M. K. B. et al. h4rm3l: A language for composable jailbreak attack synthesis. **arXiv preprint arXiv:2408.04811**, 2024.

ESMRADI, A.; YIP, D. W.; CHAN, C. F. A comprehensive survey of attack techniques, implementation, and mitigation strategies in large language models. **arXiv preprint arXiv:2312.10982**, 2023.

GANGULI, D. et al. Red teaming language models to reduce harms: Methods, scaling behaviors, and lessons learned. **arXiv preprint arXiv:2209.07858**, 2022.

GHOSH, S. et al. Aegis: Online adaptive ai content safety moderation with ensemble of llm experts. **arXiv preprint arXiv:2404.05993**, 2024.

GOODFELLOW, I. Generative adversarial networks. In: **Communications of the ACM**. [S.l.]: ACM, 2020. v. 63, n. 11, p. 139–144.

GOODFELLOW, I. et al. Generative adversarial nets. In: **Advances in neural information processing systems**. [S.l.: s.n.], 2014. p. 2672–2680.

GOODFELLOW, I. J.; SHLENS, J.; SZEGEDY, C. Explaining and harnessing adversarial examples. **arXiv preprint arXiv:1412.6572**, 2014.

GRESHAKE, K. et al. Not what you’ve signed up for: Compromising real-world llm-integrated applications with indirect prompt injection. In: **Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security**. [S.l.: s.n.], 2023. p. 79–90.

HAN, S. et al. Wildguard: Open one-stop moderation tools for safety risks, jailbreaks, and refusals of llms. **arXiv preprint arXiv:2406.18495**, 2024.

HINTON, G. E.; SALAKHUTDINOV, R. R. Reducing the dimensionality of data with neural networks. **Science**, American Association for the Advancement of Science, v. 313, n. 5786, p. 504–507, 2006.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. **Neural computation**, MIT Press, v. 9, n. 8, p. 1735–1780, 1997.

INAN, H. et al. Llama guard: Llm-based input-output safeguard for human-ai conversations. In: **arXiv preprint arXiv:2312.06674**. [s.n.], 2023. Disponível em: <<https://arxiv.org/abs/2312.06674>>.

JAGIELSKI, M. et al. Manipulating machine learning: Poisoning attacks and counter-measures for regression learning. In: IEEE. **2018 IEEE Symposium on Security and Privacy (SP)**. [S.l.], 2018. p. 19–35.

JINDAL, M. et al. SAGE: A generic framework for LLM safety evaluation. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. **Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track**. [S.l.], 2025. p. 11–33.

JOBIN, A.; IENCA, M.; VAYENA, E. The global landscape of AI ethics guidelines. **Nature Machine Intelligence**, Nature Publishing Group, v. 1, n. 9, p. 389–399, 2019.

JORDAN, M. I.; MITCHELL, T. M. Machine learning: Trends, perspectives, and prospects. **Science**, American Association for the Advancement of Science, v. 349, n. 6245, p. 255–260, 2015.

KAPLAN, J. et al. Scaling laws for neural language models. **arXiv preprint arXiv:2001.08361**, 2020.

KINGMA, D. P.; WELLING, M. Auto-encoding variational bayes. **arXiv preprint arXiv:1312.6114**, 2013.

KITCHENHAM, B.; CHARTERS, S. **Guidelines for performing systematic literature reviews in software engineering**. [S.l.], 2007.

KUMAR, P.; MISHRA, S. Robustness in large language models: A survey of mitigation strategies and evaluation metrics. **arXiv preprint arXiv:2505.18658**, May 2025.

KUMAR, R. S. S. et al. Adversarial machine learning-industry perspectives. In: IEEE. **2020 IEEE Security and Privacy Workshops (SPW)**. [S.l.], 2020. p. 75–79.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **Nature**, Nature Publishing Group, v. 521, n. 7553, p. 436–444, 2015.

LEE, J. et al. Industrial artificial intelligence for industry 4.0-based manufacturing systems. **Manufacturing letters**, Elsevier, v. 25, p. 88–91, 2020.

LI, L. et al. Salad-bench: A hierarchical and comprehensive safety benchmark for large language models. In: **Findings of the Association for Computational Linguistics: ACL 2024**. [S.l.: s.n.], 2024. ArXiv:2402.05044.

LIANG, P. et al. Helm safety: Towards standardized safety evaluations of language models. **arXiv preprint**, 2023.

LIU, X. et al. AutoDAN-Turbo: A lifelong agent for strategy self-exploration to jailbreak LLMs. In: **The Thirteenth International Conference on Learning Representations (ICLR)**. [S.l.: s.n.], 2025. Spotlight.

MANNING, C. D.; SCHÜTZE, H. **Foundations of statistical natural language processing**. [S.l.]: MIT press, 1999.

MARCHANT, G. E.; ALLENBY, B. R.; HERKERT, J. R. The growing gap between emerging technologies and the law. In: **The Growing Gap Between Emerging Technologies and Legal-Ethical Oversight**. [S.l.]: Springer, 2011. p. 19–33.

MARKOV, A. A. Theory of algorithms. **Israel Program for Scientific Translations**, 1971.

MAZEIKA, M. et al. Harmbench: A standardized evaluation framework for automated red teaming and robust refusal. **arXiv preprint arXiv:2402.04249**, 2024.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. **The Bulletin of Mathematical Biophysics**, Springer, v. 5, n. 4, p. 115–133, 1943.

MURPHY, K. P. **Machine learning: a probabilistic perspective**. [S.l.]: MIT Press, 2012.

OECD. **Recommendation of the Council on Artificial Intelligence**. Paris, 2019. Disponível em: <<https://legalinstruments.oecd.org/en/instruments/OECD-LEGAL-0449>>.

OPENAI. Gpt-4 technical report. **arXiv preprint arXiv:2303.08774**, 2023.

PAPERNOT, N. et al. Sok: Security and privacy in machine learning. **2018 IEEE European Symposium on Security and Privacy (EuroS&P)**, IEEE, p. 399–414, 2018.

PEREZ, F.; RIBEIRO, I. Ignore previous prompt: Attack techniques for language models. In: **arXiv preprint arXiv:2211.09527**. [S.l.: s.n.], 2022.

Pope Leo XIV. **Encyclical Letter *Magnifica Humanitas* on Safeguarding the Human Person in the Time of Artificial Intelligence**. Vatican City: Libreria Editrice Vaticana, 2026. No. 107. Disponível em: <<https://www.vatican.va/content/leo-xiv/en/encyclicals/documents/20260515-magnifica-humanitas.html>>.

QI, X. et al. Fine-tuning aligned language models compromises safety, even when users do not intend to! **arXiv preprint arXiv:2310.03693**, 2023.

RADFORD, A. et al. Learning transferable visual models from natural language supervision. **International Conference on Machine Learning**, PMLR, p. 8748–8763, 2021.

ROLLER, S. et al. Recipes for building an open-domain chatbot. **Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume**, p. 300–325, 2021.

RÖTTGER, P. et al. Xstest: A test suite for identifying exaggerated safety behaviours in large language models. **arXiv preprint arXiv:2308.01263**, 2023.

RUMELHART, D.; HINTON, G.; WILLIAMS, R. J. Learning representations by back-propagating errors. **Nature**, Nature Publishing Group, v. 323, n. 6088, p. 533–536, 1986.

RUSSELL, S. J.; NORVIG, P. **Artificial intelligence: a modern approach**. 3rd. ed. [S.l.]: Pearson Education Limited, 2010.

SCHICK, T. et al. Toolformer: Language models can teach themselves to use tools. **arXiv preprint arXiv:2302.04761**, 2023.

SCHNEIER, B. **Click Here to Kill Everybody: Security and Survival in a Hyper-connected World**. [S.l.]: WW Norton & Company, 2018.

SHEN, X. et al. "do anything now": Characterizing and evaluating in-the-wild jailbreak prompts on large language models. **arXiv preprint arXiv:2308.03825**, 2023.

SMUHA, N. A. Beyond the individual: governing AI's societal harm. **Internet Policy Review**, v. 10, n. 3, 2021.

SOLMS, R. von; NIEKERK, J. van. From information security to cyber security. **Computers & Security**, Elsevier, v. 38, p. 97–102, 2013.

SOULY, A. et al. A StrongREJECT for empty jailbreaks. In: **Advances in Neural Information Processing Systems (NeurIPS)**. [S.l.: s.n.], 2024.

SZEGEDY, C. et al. Intriguing properties of neural networks. In: **International Conference on Learning Representations**. [S.l.: s.n.], 2013.

THOPPILAN, R. et al. Lamda: Language models for dialog applications. **arXiv preprint arXiv:2201.08239**, 2022.

TRAMÈR, F. et al. Fundamental tradeoffs between invariance and sensitivity to adversarial perturbations. In: PMLR. **Proceedings of the 37th International Conference on Machine Learning**. [S.l.], 2020. p. 9561–9571. ArXiv:2002.04599.

TURING, A. M. Computing machinery and intelligence. **Mind**, Oxford University Press, v. 59, n. 236, p. 433–460, 1950.

VASSILEV, A. et al. **Adversarial Machine Learning: A Taxonomy and Terminology of Attacks and Mitigations**. [S.l.], 2025.

VASWANI, A. et al. Attention is all you need. In: **Advances in neural information processing systems**. [S.l.: s.n.], 2017. p. 5998–6008.

VEALE, M.; BORGESIU, F. Z. Demystifying the draft EU artificial intelligence act. **Computer Law Review International**, Schmidt, Dr. Otto, v. 22, n. 4, p. 97–112, 2021.

WACHTER, S.; MITTELSTADT, B. Data protection in the age of big data. **Nature Electronics**, Nature Publishing Group, v. 4, n. 1, p. 10–11, 2021.

WANG, B. et al. Decodingtrust: A comprehensive assessment of trustworthiness in gpt models. 2023. Outstanding Paper Award. arXiv:2306.11698.

WEI, A.; HAGHTALAB, N.; STEINHARDT, J. Jailbroken: How does llm safety training fail? **arXiv preprint arXiv:2307.02483**, 2023.

WHITMAN, M. E.; MATTORD, H. J. **Principles of Information Security**. 7th. ed. [S.l.]: Cengage Learning, 2021.

XIE, T. et al. SORRY-Bench: Systematically evaluating large language model safety refusal behaviors. **arXiv preprint arXiv:2406.14598**, 2024.

XU, Y. et al. Rlthf: Targeted human feedback for llm alignment. **arXiv preprint arXiv:2502.13417**, February 2025.

XU, Y. et al. Llm security and privacy: A survey. **arXiv preprint**, 2024. WARNING: arXiv ID 2402.18413 is incorrect — verify before submitting.

YAO, Y. et al. A survey on large language model (LLM) security and privacy: The good, the bad, and the ugly. **arXiv preprint arXiv:2312.02003**, 2023.

YUAN, Y. et al. Gpt-4 is too smart to be safe: Stealthy chat with llms via cipher. **arXiv preprint arXiv:2308.06463**, 2023.

ZENG, Y. et al. AIR-Bench 2024: A safety benchmark based on risk categories from regulations and policies. In: **The Thirteenth International Conference on Learning Representations (ICLR)**. [S.l.: s.n.], 2025.

ZHANG, Z. et al. Safetybench: Evaluating the safety of large language models with multiple choice questions. In: **Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics**. [S.l.: s.n.], 2024. ArXiv:2309.07045.

ZHANG, Z. et al. Who is responsible when AI fails? mapping causes, entities, and consequences of AI privacy and ethical incidents. **arXiv preprint arXiv:2401.12191**, 2024.

ZHOU, S. et al. Adversarial attacks and defenses in deep learning: From a perspective of cybersecurity. **ACM Computing Surveys**, ACM, v. 55, n. 8, 2023.

ZHOU, W.; WANG, X. et al. Easyjailbreak: A unified framework for jailbreaking large language models. **arXiv preprint arXiv:2403.12171**, 2024.

ZOU, A. et al. Universal and transferable adversarial attacks on aligned language models. **arXiv preprint arXiv:2307.15043**, 2023.

UNIVERSIDADE DO ESTADO DE SANTA CATARINA – UDESC
BIBLIOTECA UNIVERSITÁRIA
REPOSITÓRIO INSTITUCIONAL

CENTRO DE CIÊNCIAS TECNOLÓGICAS – CCT

ATESTADO DE VERSÃO FINAL

Eu, Charles Christian Miers, professor(a) do curso de Mestrado em Computação Aplicada declaro que esta é a versão final aprovada pela comissão julgadora da dissertação/tese intitulada: **“SAFEPROBE: A PYTHON TOOLKIT FOR PROBING THE SAFETY ALIGNMENT OF LARGE LANGUAGE MODELS”** de autoria do(a) acadêmico Milton Pedro Pagliuso Neto.

Joinville, 16 de Julho de 2026.*

Assinatura digital do(a) orientador(a):



Documento assinado digitalmente
CHARLES CHRISTIAN MIERS
Data: 16/07/2026 17:57:17-0300
Verifique em <https://validar.iti.gov.br>

Charles Christian Miers