Technical Report

Frontier Mitigations

FRONTIER MODEL FORUM

REPORT SERIES

Implementing Frontier AI Frameworks

DATE

June 30, 2025

About the Report

This is the third in a **series of technical reports** on frontier AI frameworks that examine how the frameworks can be used effectively across different organizational contexts. The series intends to provide detailed insight into key components of these frameworks, incorporating lessons from early adopters while acknowledging areas where best practices continue to evolve.

Executive Summary

Frontier mitigations are protective measures implemented on frontier models, with the goal of reducing the risk of potential high-severity harms, especially those related to national security and public safety, that could arise from their advanced capabilities.

This report discusses emerging industry practices for implementing and assessing frontier mitigations. It focuses on mitigations for managing risks in three primary domains: chemical, biological, radiological and nuclear (CBRN) information threats, where AI could be misused to lower barriers to developing weapons of mass destruction; advanced cyber threats, where AI could be misused to enable sophisticated attacks against critical infrastructure; and advanced autonomous behavior threats, including AI systems that could recursively self-improve or conduct autonomous AI research and development (R&D).

Given the nascent state of frontier mitigations, this report describes the range of controls and mitigation strategies being employed or researched by Frontier Model Forum (FMF) members and documents the known limitations of these approaches. The report focuses on model- and system-level mitigations, and therefore does not cover broader controls to reduce risk, such as organizational security, risk governance, and safety culture. Some mitigations described here, such as the deployment of classifiers or the monitoring of end users, may also be relevant to downstream developers configuring the deployments of their own AI systems.

The following types of mitigations are common:

- **Capability Limitation Mitigations**: Approaches that aim to prevent models from possessing knowledge or abilities that could enable harm. These methods alter the model's weights or training process. Examples include data filtering during training, targeted unlearning of specific knowledge after training, and model distillation. However, the effectiveness of these mitigations is an active area of research, and they can be circumvented if dual-use knowledge is added to the model during inference or fine-tuning. See Section 2 for implementation details and further examples.
- **Behavioral Alignment Mitigations**: Approaches that seek to prevent a model's potentially dangerous capabilities from being elicited by shaping its responses to human requests and its autonomous decision-making processes. These methods aim to train models to refuse inappropriate user requests and maintain appropriate goals and behaviors when operating autonomously. Examples include supervised fine-tuning on refusal examples, reinforcement learning from human feedback, and constitutional AI approaches. However, whether these methods can reliably address challenges related to robustness, training signal design, and goal learning and persistence remains an open question. See Section 3 for implementation details and further examples.
- **Detection and Intervention Mitigations**: Approaches that rely on automated methods to detect model usage (e.g., inputs and outputs) that may give rise to undesired behavior. These methods can be applied to both misuse risks and misalignment risks. Examples include content filtering systems, large language model (LLM)-based prompted classifiers, custom-trained classifier models, linear classifier probes, and intervention mechanisms that respond proportionally to detected concerns. See Section 4 for implementation details and further examples.
- **Access Control Mitigations**: Approaches that govern who can use a model, what capabilities they can access, and how the model can interact with external systems. These methods establish boundaries that determine the conditions under which model capabilities can be utilized. Examples include user verification protocols, tiered access levels, staged deployment, and model permissions and sandboxing. See Section 5 for implementation details and further examples.
- **Supporting Ecosystem Mitigations**: Approaches where developers provide information, tools, and capabilities that enable other actors – governments, organizations, and civil society – to implement effective defenses against AI-enabled threats. Rather than directly controlling societal defenses, developers contribute by sharing resources that strengthen the broader defensive ecosystem. Examples include information sharing and documentation, supporting defensive systems and research, and establishing reporting and early warning systems. See Section 6 for implementation details and further examples.

In practice, the appropriateness and efficacy of any specific mitigation measure may vary depending on the nature of the model and the manner in which it is made available to the public. For example, open-weights model providers, who make models available for downstream developer use and customization, may have different practices compared to providers that offer models via an API. Since the application of certain mitigations may come with trade-offs that impact model and system usefulness, developers often employ distinct mitigation approaches influenced by variations such as the deployment context, threat actor capabilities, technical constraints, organizational resources, and target levels of residual risk. They typically implement multiple mitigations as complementary layers of protection rather than relying on any single

approach.

Following the implementation of mitigations, developers conduct **mitigation effectiveness assessments** to verify whether implemented measures achieve their intended risk reduction. These assessments examine how mitigations perform under various conditions, including adversarial scenarios, and identify areas for improvement.

Developers evaluate mitigations both individually and as integrated systems, considering factors such as ease of circumvention, redundancy across multiple measures, and performance across different deployment contexts. Testing controls individually enables more precise identification of component-specific weaknesses, earlier issue detection, and targeted improvements. Testing controls in combination can give a clearer picture of how the combined system will behave under realistic deployment conditions or reveal emergent vulnerabilities from components that function correctly in isolation. Possible approaches to testing controls include internal safety teams, external red-teams, researcher access programs, bug bounty programs, and red team vs blue team exercises. These mitigation effectiveness assessment methods are discussed more in Section 7.

Some developers and third-party researchers have also begun exploring structured documentation approaches for their mitigation assessments, inspired by safety case approaches in other industries, and structured approaches for translating safeguard evaluations into residual risk estimates, considering the individual controls, their effectiveness in combination, and the surrounding infrastructure security.

This report outlines current approaches to frontier mitigations, but as capabilities advance and our understanding of effective safeguards improves, developer approaches will be updated accordingly. Areas of continued work include developing more robust safety mechanisms that resist circumvention attempts, exploring new approaches to capability limitation and behavioral alignment that remain effective as models scale, and establishing best practices for implementing controls across different deployment contexts. See more in Section 8.

Section 1

Overview of Frontier Mitigations

ROADMAP

1.1 Purpose and Scope

1.2 Elements of Effective Frontier Mitigations

1.3 Common Frontier Mitigation Approaches

1.4 Relationship to Safety and Security Frameworks

1.1 Purpose and Scope

Frontier mitigations are protective measures implemented on frontier models, with the goal of reducing the risk of potential high-severity harms, especially those related to national security and public safety, that could arise from their advanced capabilities. This report focuses on model- and system-level mitigations, and therefore does not cover broader controls to reduce risk, such as organizational security, risk governance, and safety culture.

This report discusses emerging industry practices for implementing and assessing frontier mitigations. It focuses on mitigations for managing risks in three primary domains: chemical, biological, radiological and nuclear (CBRN) information threats, where AI could be misused to lower barriers to developing weapons of mass destruction; advanced cyber threats, where AI could be misused to enable sophisticated attacks against critical infrastructure; and advanced autonomous behavior threats, including AI systems that could recursively self-improve or conduct autonomous AI research and development (R&D). These domains represent areas where frontier AI capabilities could lead to severe, large-scale harms if developed or deployed without adequate safeguards.

Where the report references developer practices, it is primarily referring to the current practices of Frontier Model Forum (FMF) members. Frontier mitigations and their assessment methods are an active area of research and development. As frontier technology continues to evolve, we anticipate that new mitigations will be developed and others will become less effective or require updating. The report does not cover the following topics, although they are or will be covered in other technical reports: identifying potential high-severity risks from frontier AI and establishing related thresholds, assessing whether models have capabilities that could increase the risk of these high-severity harms, or conducting third-party assessments.

1.2 Elements of Effective Frontier Mitigations

Frontier mitigations can be technical – modifying model architecture, training processes, or deployment infrastructure – or administrative – establishing governance frameworks, policies, and decision-making processes. Frontier mitigations can operate at various levels, from changes to the model itself to broader system-level or infrastructure-level protections.

Mitigation Effectiveness Assessments verify whether implemented measures achieve their intended risk reduction. These assessments examine how mitigations perform under various conditions, including adversarial scenarios, and identify areas for improvement. Developers evaluate mitigations both individually and as integrated systems, considering factors such as ease of circumvention, redundancy across multiple measures, and performance across different deployment contexts.

Underpinning both elements is **Threat Modeling** – the process of identifying and analyzing potential threats to public safety and security, typically in consultation with domain experts. This process is discussed further in an FMF report on risk taxonomies and thresholds. Threat modeling informs both the selection of appropriate mitigations and their effectiveness assessment. For example, understanding threat actor capabilities and resources helps identify which mitigations might be vulnerable to circumvention, enabling developers to implement stronger protections where needed.

1.3 Common Frontier Mitigations Approaches

Developers employ distinct mitigation approaches influenced by variations such as the deployment context, threat actor capabilities, technical constraints, organizational resources, and target levels of residual risk. They typically implement multiple mitigations as complementary layers of protection rather than relying on any single approach. This layered defense allows flexibility – models with stronger behavioral alignment might need fewer access restrictions, while open-weight releases might implicate different approaches compared to API deployments. Developers calibrate their mitigations based on threat models, balancing risk reduction against preserving beneficial capabilities. Implementation requirements vary significantly in time and resources, and so some organizations use risk forecasts to plan mitigation timelines.

The following types of mitigations are common:

- **Capability Limitation Mitigations**: Approaches that aim to prevent models from possessing knowledge or abilities that could enable harm. These methods alter the model's weights or training process. Examples include data filtering during training, targeted unlearning of specific knowledge after training, and model distillation. However, the effectiveness of these mitigations is an active area of research, and they can be circumvented if dual-use knowledge is added to the model during inference or fine-tuning. See Section 2 for implementation details and further examples.

- **Behavioral Alignment Mitigations**: Approaches that seek to prevent a model's potentially dangerous capabilities from being elicited by shaping a model's responses to human requests and its autonomous decision-making processes. These methods aim to train models to refuse inappropriate user requests and maintain appropriate goals and behaviors when operating autonomously. Examples include supervised fine-tuning on refusal examples, reinforcement learning from human feedback, and constitutional AI approaches. However, whether these methods can reliably address challenges related to robustness, training signal design, and goal learning and persistence remains an

open question. See Section 3 for implementation details and further examples.

- **Detection and Intervention Mitigations**: Approaches that rely on automated methods to detect model usage (e.g., inputs and outputs) that may give rise to undesired behavior. These methods can be applied to both misuse risks and misalignment risks. Examples include content filtering systems, large language model (LLM)-based prompted classifiers, custom-trained classifier models, linear classifier probes, and intervention mechanisms that respond proportionally to detected concerns. See Section 4 for implementation details and further examples.
- **Access Control Mitigations**: Approaches that govern who can use a model, what capabilities they can access, and how the model can interact with external systems. These methods establish boundaries that determine the conditions under which model capabilities can be utilized. Examples include user verification protocols, tiered access levels, staged deployment, and model permissions and sandboxing. See Section 5 for implementation details and further examples.
- **Supporting Ecosystem Mitigations**: Approaches where developers provide information, tools, and capabilities that enable other actors – governments, organizations, and civil society – to implement effective defenses against AI-enabled threats. Rather than directly controlling societal defenses, developers contribute by sharing resources that strengthen the broader defensive ecosystem. Examples include information sharing and documentation, supporting defensive systems and research, and establishing reporting and early warning systems. See Section 7 for implementation details and further examples.

This report presents current, emerging and experimental mitigation practices rather than an exhaustive list. Other equally valid approaches exist, and not all described mitigations are suitable for every model or deployment. Each section also includes promising future directions in that mitigation category.

1.4 Relationship to Safety and Security Frameworks

Frontier Model Forum (FMF) members implement frontier AI frameworks that integrate capability assessments, risk thresholds, and mitigations into a structured risk management process. These frameworks typically use a two-stage system:

1. **Capability Assessments**: The process begins with **frontier capability assessments** that evaluate whether a model crosses any **enabling capability thresholds** – possessing abilities that could enable severe harms, such as substantially assisting with chemical, biological, radiological, and nuclear (CBRN) weapons development. When a model crosses such a threshold, the framework triggers mitigation and risk evaluation requirements.

2. **Mitigation Assessments**: For models that cross enabling capability thresholds, developers apply appropriate frontier mitigations, such as the approaches described in this report. Developers then **evaluate their effectiveness** and whether the residual risk is acceptable for proceeding with further training or deployment. Whether a model meets this **acceptable development or deployment threshold** will depend on factors including the deployment context and the broader risk landscape.

This approach allows developers to scale safety measures with model capabilities, applying more stringent mitigations when models demonstrate abilities that could pose significant risks.

Section 2

Capability Limitation Mitigations

ROADMAP

2.1 Data Filtering

2.2 Exploratory Methods

Capability limitation mitigations aim to prevent models from possessing knowledge or abilities that could enable harm. These methods alter the model's weights or training process, so that it cannot assist with harmful actions when prompted by humans or autonomously pursue harmful objectives. However, the effectiveness of these mitigations is an active area of research, and they can currently be circumvented if dual-use knowledge (knowledge that has both benign and harmful applications) is added in the context window during inference or fine-tuning.

2.1 Data Filtering

Data filtering involves removing content from training datasets that could lead to dual-use or potentially harmful capabilities. Developers can use several methods: automated classifiers to identify and remove content related to weapons development, detailed attack methodologies, or other high-risk domains; keyword-based filters to exclude documents containing specific terminology or instructions of concern; and machine learning models trained to recognize subtle patterns in content that might contribute to dangerous capabilities.

While data filtering can reduce certain risk-relevant capabilities, it faces limitations:

1. Much scientific and technical knowledge has dual-use applications – essential for beneficial purposes but potentially enabling harm. For example, training data about biological sequences and pathogen research enables models to assist with vaccine development and disease understanding, but these same capabilities could potentially help malicious actors modify dangerous pathogens. Similarly, cybersecurity knowledge allows models to help defend systems, but some subset of this knowledge could also be applied by malicious

actors to develop attacks. Removing all such content would severely degrade model usefulness across legitimate scientific, medical, and security applications.

2. Capabilities of concern can also emerge from combinations of seemingly benign training data through unexpected inferences. Models may develop the ability to assist with dangerous tasks by connecting disparate pieces of information, even when explicitly problematic content has been filtered. For instance, a model might combine general chemistry knowledge with unrelated optimization techniques to assist with potentially harmful synthesis, despite neither domain being inherently problematic on its own.

These limitations mean data filtering should be combined with other mitigation approaches rather than used in isolation. Further research is needed to understand which harmful outputs can be effectively prevented through data filtering alone versus those requiring additional safeguards.

2.2 Exploratory Methods

Beyond data filtering, researchers are investigating additional capability limitation approaches. However, these methods face technical challenges, and their effectiveness remains uncertain.

- **Model distillation** could create specialized versions of frontier models with capabilities limited to specific domains. For example, a model could excel at medical diagnosis while lacking knowledge needed for biological weapons development. While the capability limitations may be more fundamental than post-hoc safety training, it remains unclear how effectively this approach prevents harmful capabilities from being reconstructed. Additionally, multiple specialized models would be needed to cover various use cases, increasing development and maintenance costs.

- **Targeted unlearning** attempts to remove specific dangerous capabilities from models after initial training, offering a more precise alternative to full retraining. Possible approaches include fine-tuning on datasets to overwrite specific knowledge while preserving general capabilities, or modifying how models internally structure and access particular information. However, these methods may be reversible with relatively modest effort – restoring "unlearned" capabilities through targeted fine-tuning with small datasets. Models may also regenerate removed knowledge by inferring from adjacent information that remains accessible.

While research continues on these approaches, developers currently rely more heavily on post-deployment mitigations that can be more reliably implemented and assessed.

Section 3

Behavioral Alignment Mitigations

ROADMAP

3.1 Current Alignment Methods

3.2 Promising Research Directions

Behavioral alignment mitigations shape how models respond to human requests and make autonomous decisions, aiming to prevent dangerous capabilities from being elicited or expressed while maintaining helpful behavior. These methods focus on training models to refuse inappropriate requests and maintain aligned goals during autonomous operation.

3.1 Current Alignment Methods

Pre-trained models possess broad capabilities but lack built-in safety protocols, potentially generating harmful outputs or failing to follow instructions. "Post-training" processes steer model behavior to achieve instruction adherence, policy compliance, and other desirable response properties. Common post-training methods include:

- **Supervised Fine-Tuning (SFT)**: Developers curate datasets of desired model behaviors and fine-tune models to match these examples. Datasets include refusal examples (such as declining to provide bomb-making instructions) and helpful yet harmless responses. SFT directly teaches models specific behavioral patterns through imitation learning.

- **Reinforcement Learning from Human Feedback (RLHF)**: Developers use human preferences between different model outputs to questions as a reward signal. They then use reinforcement learning to optimize models for these reward signals.

- **Reinforcement Learning from AI-assisted Feedback (RLAIF)**: Methods like Anthropic's Constitutional AI, OpenAI's deliberative alignment, and the broader category of AI-assisted feedback, including RLAIF, use AI systems to generate training feedback based on predefined principles or constitutions

(such as OpenAI's model spec), and have gained traction as a scalable alternative to purely human-generated feedback.

Modern alignment training pipelines typically combine these methods. These approaches can also steer the model towards prioritizing certain types of requests over others (such as system-level safety specifications over others). Reward signals for agentic AI systems could also consist of more complex and real-world tasks, such as writing safe software code.

However, behavioral alignment approaches have several limitations, open questions, and possible challenges:

1. **Robustness Challenges**: Current safety training methods modify surface-level behaviors without altering underlying model capabilities. Research shows that harmful fine-tuning can rapidly undo alignment post-training with surprisingly few examples – sometimes just dozens of harmful input-output pairs. Adversarial prompts ("jailbreaks") can bypass alignment post-training – models trained to refuse direct harmful requests may still comply when those requests are adversarially rephrased, decomposed into steps, or embedded in different contexts. Developers can continuously patch new vulnerabilities; however, in attempting to correct for this, a different problem can occur with "over-refusal," where models become excessively cautious and decline legitimate requests like medical questions.

2. **Training Signal Learning and/or Design Challenges**: Behavioral alignment requires translating human values into training objectives, but this translation introduces challenges. Reward misspecification and/or "reward hacking" could occur, with models exploiting flaws in reward signals, such as generating unnecessarily verbose responses that score well but provide little value, or appealing to evaluator biases rather than producing useful outputs. Models might also learn different objectives than intended, even when performing well on a well-designed training signal, and could potentially then "fake alignment" to avoid further modification. These different objectives could potentially emerge from the model learning proxy goals that happen to achieve good training performance but generalize in undesired ways (also known as "goal misgeneralization").

3. **Understanding and Measurement Gaps**: Although there is some early work in the area, the field currently lacks reliable methods to assess alignment effectiveness, making it difficult to determine how well current techniques will scale to more advanced systems.

These limitations highlight the need for continued research into the significance of these challenges and methods that can scale reliably with advancing model capabilities.

3.2 Promising Research Directions

The following research directions show promise for improving behavioral alignment mitigations and/or our understanding of their effectiveness:

- **Adversarial Training**: Developers can improve processes to search for failure modes through human and automated red-teaming methods, and analysis of real-world jailbreaks, and then incorporate fixes into training. While this can create an ongoing cat-and-mouse dynamic and may fail to address certain novel attacks, it acts as a baseline defense.

- **Fine-tuning Security Controls**: When offering fine-tuning capabilities, developers could screen users or use classifiers to detect datasets with harmful content patterns, and/or suspicious instructions designed to compromise safety guardrails. These systems could be configured to filter problematic data, flag borderline cases for human review, or reject entire datasets that appear malicious. Developers could also conduct comparative assessments before and after fine-tuning to detect degradation in safety properties or suspicious capability enhancements.

- **Scalable Oversight**: Developing methods where AI systems help evaluate other AI systems could leverage the principle that evaluation is typically easier than generation. Possible research directions in this area include recursive reward modeling (using AI assistance to evaluate increasingly complex behaviors) and debate (having models argue different positions to expose flaws). These approaches aim to maintain meaningful human oversight even as models exceed human capabilities in specific domains.

- **Mechanistic Interpretability-Guided Interventions**: Developing a mechanistic understanding of model internals could enable more targeted safety modifications.

While progress in these areas could significantly improve model safety, the relative nascent nature of the research and uncertainty about its effectiveness mean developers may use complementary approaches such as detection and intervention, and access controls, as discussed in later sections.

Section 4

Detection and Intervention Mitigations

ROADMAP

4.1 LLM-based Prompted Classifiers

4.2 Custom-trained Classifier Models

4.3 Linear Probes

4.4 Static Analysis Tools

4.5 Manual Human Review

4.6 Promising Research Directions

Detection and intervention mitigations rely on automated methods to detect model usage (e.g., inputs and outputs) that may give rise to undesired behavior. These detection methods are paired with intervention mechanisms that respond proportionally to detected concerns. The subsections below describe various **detection methods**, while this introduction outlines how these methods are combined with different **intervention methods.** Detection and intervention methods can be applied to both misuse risks and misalignment risks, where models might pursue unintended goals or exhibit deceptive behavior autonomously.

Detection systems can analyze many different inputs to identify potential concerns, including: user queries, model responses, conversation history, fine-tuning data, background information like past user interactions, activations, or chain-of-thought (as described in OpenAI's o1 system card). Developers, including downstream adopters of open-weight models, can also combine multiple detection methods into **orchestration layers** to maximize coverage while compensating for individual method limitations. These orchestration layers coordinate different detection approaches – from simple classifiers to complex probes – and route detected issues to appropriate interventions. For agent misalignment risks, systems like Meta's Llama Firewall demonstrate how multiple detection methods can work together to provide more robust protection.

Once concerning behavior is detected, interventions can be implemented based on the severity and nature of the risk. Common interventions include blocking or modifying model responses before they reach users, terminating user sessions, flagging content for human review, or, in severe cases, escalating to law enforcement. Developers typically implement escalating response protocols that match intervention intensity to threat severity.

4.1 LLM-based Prompted Classifiers

Language models can be prompted to act as classifiers on inputs to and outputs of the model. The classifier large language model (LLM) can be another instance of the same model or a separate model optimized for classification tasks. For example, it has been demonstrated that prompting an LLM with simple instructions like "Does the text contain harmful content? Respond with 'Yes, this is harmful' or 'No, this is not harmful'" can achieve high accuracy and effectively reduce attack success. LLM-based classifiers could also monitor for signs of misalignment in model outputs, particularly in autonomous agent settings. In some cases, it is also possible to generate probability scores rather than binary classifications.

These LLM-based prompted classifiers are relatively simple to set up and deploy compared to other monitoring approaches, and can be easily created and customized from open-weight models. This flexibility allows developers to adjust classification thresholds based on their risk tolerance and deployment context. However, the more inputs that require assessment, the more compute and latency costs are incurred. Although costs can be reduced by using smaller models to monitor, this may lead to decreased accuracy. Additionally, LLM-based prompted classifiers are prone to circumvention if malicious prompts are separated into seemingly benign substeps or distributed across different accounts.

4.2 Custom-trained Classifier Models

Language models can be trained via fine tuning to classify inputs and outputs from the model, producing probability scores that indicate whether content violates safety policies, contains potentially harmful material, or exhibits misalignment-related behavior. Examples that demonstrate this approach include: Google's ShieldGemma, which provides a suite of models from 2 billion to 27 billion parameters that achieve state-of-the-art performance with harm-type specific predictions; Meta's Llama Guard, which uses Llama Guard 4 with 12 billion model instruction-tuned parameters for multi-class classification with customizable safety taxonomies; and Anthropic's Constitutional Classifiers, which leverage constitution-guided synthetic data generation to train classifiers that successfully defended against universal jailbreaks across thousands of hours of human red teaming. As demonstrated in the Constitutional Classifiers example, developers can guide training using explicit rules-(a "constitution") defining permissible and restricted content. Explicitly defining both harmful and harmless categories in this way can help produce more nuanced training data, enabling the classifier to learn appropriate boundaries more effectively.

Compared to prompting a language model to act as a classifier, custom-trained classifier models can reduce the cost and latency involved in scaling to many inputs. This approach also enables real-time vetting of the model's response as it is being generated, allowing immediate intervention if harmful content is detected. The drawbacks include complexity and development time compared to methods like linear classifier probes (see below). Custom-trained classifier models also tend to be more expensive and introduce more latency compared to probes at runtime, though often less than LLM-based prompted classifiers.

4.3 Linear Probes

Linear probes, also referred to as "probes", are simple classifiers trained to predict specific properties based on a model's activations. These activations are the intermediate computational states produced as the model processes input and represent the internal calculations or processing taking place inside a model

while generating an output. Probes can be trained to check these activations for patterns related to undesirable behavior, such as generating harmful content or being deceptive.

Probes offer several advantages over other monitoring methods: they require less compute than running a prompted language model and introduce minimal latency to the model's response time; their simplicity and low computational cost also make them relatively easy to retrain with new data or adjust during development, such as changing detection thresholds to be more or less strict; and they can potentially identify problematic internal states even when the input or output appears benign, which may be particularly useful for misalignment detection.

The limitations of probes include generally needing to be retrained for each new model version, although this retraining process can often be relatively quick. Probes may also face generalization issues in detecting different or more complex examples encountered in real-world use. Lastly, implementation choices, such as which layers of the model to probe and how to aggregate probe scores across the tokens output by the model (e.g., averaging across all tokens, focusing on specific critical areas of the response, or taking the highest score), involve tradeoffs that affect overall accuracy and sensitivity.

4.4 Static Analysis Tools

Static tooling can also be deployed to increase the robustness of LLM outputs. For example, CodeShield, part of Meta's PurpleLlama project, is a static analysis engine that detects insecure code patterns in LLM-generated code, identifying vulnerabilities like weak cryptographic functions and other Common Weakness Enumeration or CWE classified security issues. Similarly, regular expressions (regex) can be used for pattern matching to validate output formats or detect prohibited content patterns.

These tools typically have lower latency compared to AI-powered tooling and can provide consistent, deterministic results for specific types of checks. However, static tools may not be comprehensive and can miss nuanced or context-specific vulnerabilities, and so they work best when combined with other detection methods.

4.5 Manual Human Review

In some cases, humans can review outputs that have been flagged for abuse as potentially problematic by automated classifiers. Microsoft's Azure OpenAI Service demonstrates this approach, where authorized Microsoft employees assess content flagged through automated classification when confidence thresholds are not met or in complex contexts.

Manual human review is potentially less exploitable compared to other monitoring approaches in the near term, as humans may be better equipped to judge complex or ambiguous cases that fall into grey areas of policy. This makes human review valuable for high-stakes decisions and for calibrating automated systems. However, manual review is inherently slow and difficult to scale compared to automated methods. There are also privacy considerations in giving human reviewers access to user inputs or potentially sensitive AI outputs.

4.6 Promising Research Directions

Current detection and intervention systems face challenges that will likely intensify as models become more capable. Research efforts are focused on addressing these limitations through several complementary approaches.

- **Improving Recall and Precision**: Detection systems face an inherent tradeoff between catching concerning behavior (recall) and avoiding false positives that degrade user experience (precision). High-recall systems often over-refuse harmless queries, increase latency, or impose other costs on deployment. Promising research directions include developing more nuanced detection methods that can distinguish edge cases more accurately, creating adaptive systems that adjust sensitivity based on context and user history, and exploring ensemble approaches that combine multiple detection methods to achieve better recall-precision tradeoffs without sacrificing usability.

- **Chain-of-Thought Monitoring and Faithfulness**: Monitoring a model's externalized reasoning process offers a promising approach for detecting concerning intent before it manifests in outputs. For example, OpenAI's o1 system card describes their chain-of-thought deception monitoring setup, while Meta has openly released 'AlignmentCheck,' a tool designed to detect instances in a model's chain of thought where prompt injection might misalign agent behavior from a user's request. Google has also developed a framework for assessing whether models possess the stealth and situational awareness capabilities necessary for evading monitoring. However, as models become more sophisticated, they may learn to produce reasoning that appears benign while concealing harmful intent, potentially through reward hacking (exploiting flaws in reward signals to achieve high scores without genuinely safe behavior) during training. Establishing chain-of-thought faithfulness and effective monitoring will therefore involve multiple considerations, including keeping reasoning visible and legible to humans for auditing purposes, avoiding training approaches that might incentivize models to conceal their true reasoning, and developing methods to detect when externalized thoughts don't match actual behavior.

- **Privacy-Preserving Monitoring and Retrospective Analysis**: Maintaining logs for retrospective analysis can be a valuable and cost-effective enabler of other safety interventions, allowing developers to identify patterns that real-time detection systems miss and understand classifier blind spots through historical analysis. This becomes particularly important as developers face unfamiliar risks from sophisticated autonomous behavior in complex environments. However, comprehensive logging raises significant privacy concerns for users who may not want their interactions retained. As models gain more autonomous capabilities and broader access to external systems, finding approaches that provide sufficient visibility for safety while respecting user privacy represents an important area for future research and careful consideration of tradeoffs.

- **Interpretability-Based Monitoring**: Surface-level monitoring of inputs and outputs may miss concerning behavior that manifests only in a model's internal computations, especially as models develop more complex reasoning capabilities. Advances in mechanistic interpretability and internal activation analysis aim to detect concerning patterns directly from model internals, even when external behavior appears benign. This includes developing methods to identify and monitor safety-relevant circuits within models, creating real-time activation monitoring systems that can flag unusual internal states.

Section 5

Access Control Mitigations

ROADMAP

5.1 Access Control Frameworks

5.2 Staged Deployment

5.3 Model Permissions and Sandboxing

Access Control Mitigations govern who can use a model, what capabilities they can access, and how the model can interact with external systems. Approaches include user verification protocols, tiered access levels, geographic restrictions, sandboxed execution environments, and restrictions on model permissions. These mitigations can help reduce risk by preventing unauthorized actors from accessing advanced capabilities, limiting which features are available to different user groups, and constraining how models can interact with external systems.

5.1 Access Control Frameworks

While developers typically implement mitigations for general access deployments, they may complement this with additional access controls when providing versions with modified or reduced mitigations for specialized use cases. For example, a developer might provide safety researchers with access to a model version with reduced refusal training to enable more comprehensive risk assessments, or provide verified medical researchers with a model that has fewer restrictions on discussing pathogen characteristics for legitimate vaccine development work. Such provisions may not be necessary if standard safeguards are sufficient for all intended users and use cases.

In such cases, developers may implement access restrictions so that only verified and authorized users can interact with models with these modified mitigations. To determine appropriate access levels, developers might define "acceptable use policies" based on threat modeling and cost-benefit analysis. Such frameworks can then serve as a guide for reviewing users' intended use cases, validating user identity and trustworthiness, and determining the appropriate access given the required modifications to safeguards.

When designing access frameworks for modified mitigations, developers might consider several factors. Developers might consider implementing robust identity verification processes, making it difficult for malicious actors to create multiple accounts to circumvent restrictions, or requiring users with enhanced access to implement security best practices, reducing the probability that these groups become vectors through which malicious actors could access less safeguarded capabilities. Access controls might also be combined with monitoring systems to detect misuse, particularly for users with elevated permissions where standard safeguards have been relaxed. Monitoring systems could detect unusual activity patterns such as access from unexpected geographic locations, suspicious API call patterns, or attempts to circumvent restrictions. However, monitoring approaches would need to be calibrated to the use case – for instance, when users are working with sensitive or classified information, privacy-preserving access controls without detailed monitoring might be more appropriate.

This tiered approach could allow developers to enable powerful capabilities for beneficial applications while maintaining appropriate protective measures. However, implementing such frameworks involves important tradeoffs – balancing security requirements against usability for legitimate research, and weighing identity verification needs against privacy considerations.

5.2 Staged Deployment

Developers may implement staged rollouts for powerful new models, starting with highly controlled environments and gradually expanding access as controls are validated. Initial deployment could involve small groups of external users or research partners operating under strict monitoring agreements, allowing developers to observe model behavior and identify potential risks before broader release.

Subsequent stages could then expand systematically: For example, deploying first to verified commercial customers with specific use cases, then to researchers and academics with appropriate credentials, and eventually to the broader public with appropriate safeguards. Each stage provides data about usage patterns and potential risks that inform the next phase of deployment.

5.3 Model Permissions and Sandboxing

Beyond controlling user access, developers implement restrictions on what models themselves can do within their operating environment. For example, sandboxing can isolate model execution from sensitive systems through contained environments with limited network access, restricted file system permissions, API rate limits, and disabled access to external tools for high-risk operations. Permission systems can further control model capabilities by requiring human approval for code execution, limiting database access based on user authorization, requiring explicit consent before accessing sensitive data, and implementing time limits on autonomous operation.

These controls are especially relevant to managing risks from potential model misalignment or autonomous behavior, and can mean that even if other safeguards fail, the model's ability to cause harm remains limited by its operating environment.

Section 6

Supporting Ecosystem Mitigations

ROADMAP

6.1 Information Sharing and Documentation

6.2 Supportive Defensive Systems and Research

6.3 Reporting and Early Warning Systems

Supporting ecosystem mitigations involves developers providing information, tools, and capabilities that enable other actors – governments, organizations, and civil society – to implement effective defenses against AI-enabled threats. While they may often not be the primary actor for societal defenses, developers can contribute by sharing resources that strengthen the broader defensive ecosystem.

6.1 Information Sharing and Documentation

Developers can enhance AI ecosystem defenses through voluntary information sharing. This includes sharing threat models and risk assessments that help recipients understand potential threat vectors and/or inform defensive strategies. This information sharing requires balancing transparency with security and legal considerations, such as providing sufficient detail to enable defenses without creating roadmaps for malicious actors.

6.2 Supporting Defensive Systems and Research

Developers, other actors in the AI ecosystem, and society broadly can develop or support the development of systems specifically designed to strengthen defensive capabilities, potentially including the use of frontier models. In the biological domain, this could include supporting improved pathogen surveillance using data sources like wastewater monitoring and open-source health reporting, investment into improved personal protective equipment resources, or improved DNA synthesis screening to detect potentially dangerous orders. For cybersecurity, developers could support the development of vulnerability detection tools for critical infrastructure. Developers could also support research into detecting and preventing autonomous misalignment, such as tools for monitoring goal drift or unexpected model behaviors.

6.3 Reporting and Early Warning Systems

Developers can establish mechanisms for rapid threat detection and response. This includes contributing to early warning systems by reporting novel threats or concerning behaviors to relevant authorities, establishing clear reporting channels for security researchers and incident responders to communicate with AI developers, and developing internal thresholds for escalating concerns to appropriate government agencies. These systems enable faster response to emerging threats across the ecosystem.

Section 7

Effectiveness Assessments for Frontier Mitigations

ROADMAP

7.1 Testing Individual Controls

7.2 Testing Controls in Combination

7.3 Modeling Residual Risk

7.4 Documenting Mitigation Assessments

Following the implementation of mitigations, it is important for developers to verify that the mitigation measures effectively reduce risks to acceptable levels under realistic conditions.

7.1 Testing Individual Controls

Developers may test specific technical properties of individual safeguards, such as precision and recall of classifier models, jailbreak detection rates of detection systems, or effectiveness of behavioral guardrails against prompt variations. Developers may also use generative models to create diverse test cases, helping to identify coverage gaps. Testing individual controls enables more precise identification of component-specific weaknesses, earlier issue detection, and targeted improvements.

7.2 Testing Controls in Combination

Developers may test mitigations in combination to evaluate how safeguards function as an integrated system. This can give a clearer picture of how the combined system will behave under realistic deployment conditions or reveal emergent vulnerabilities from components that function correctly in isolation. Possible approaches include:

- **Internal Safety Teams**: Teams within the organization conduct structured testing using knowledge of model architecture and training methods. This approach benefits from institutional knowledge but may suffer from blind spots due to organizational biases.
- **External Red-Teams**: Specialized firms that conduct adversarial testing with domain-specific expertise (e.g., biosecurity, cybersecurity), using established frameworks, and bringing experience from testing multiple models across

organizations.

- **Researcher Access Programs**: Selected academic and non-profit researchers receive privileged access to conduct independent analyses, focusing on specific research questions in emerging areas like interpretability or novel attack vectors, and publish findings after responsible disclosure periods.
- **Bug Bounty Programs**: Open programs financially incentivize external researchers to discover and report vulnerabilities (e.g., novel jailbreaking techniques) through predetermined reward structures, with clear scope definitions and verification procedures to manage submissions.
- **Red Team vs Blue Team Exercises**: Developers may run structured adversarial exercises where red teams attempt to exploit vulnerabilities in mitigation systems while blue teams defend and monitor for attacks. These exercises can stress-test monitoring systems against increasingly sophisticated model behaviors, attack strategies, and autonomous misalignment scenarios.
- **System Safety Analysis**: Although not yet common practice, system safety frameworks like System-Theoretic Process Analysis (STPA) may also be valuable for identifying hazards that emerge from system component interactions rather than from individual component failures.
- **Ongoing Deployment Monitoring**: Developers may track mitigation effectiveness throughout deployment, as safeguard efficacy can degrade when attackers develop new circumvention methods. Operational metrics might include detection system performance, attack sophistication trends, and intervention success rates, enabling timely updates when protections require reinforcement.

Across these testing methods, findings may depend significantly on how closely testing conditions mirror real-world deployment scenarios, including financial incentives, quality-of-service constraints like rate limits or latency, and available tooling. Because of this, many developers provide evaluation teams with equal or greater affordances than what they expect adversaries to have in practice, including privileged access, like model weights or increased rate limits, to establish conservative risk estimates – similar to how penetration testing programs in cybersecurity often grant initial access to certain network layers. Accordingly, developers may vet red teams more extensively or put access restrictions in place to address the security concerns that arise with external access to a model with equal or greater affordances than what is available to adversaries.

7.3 Modeling Residual Risk

Some developers and third-party researchers have begun exploring structured approaches for translating safeguard evaluations into residual risk estimates, considering the individual controls, their effectiveness in combination, and the surrounding infrastructure security. For example, creating a model of how effectively the mitigation system reduces risks compared to an unmitigated baseline, and using sensitivity analysis to identify which factors most significantly impact residual risk. These models may integrate information including:

1. Data about safeguard robustness – how consistently safety measures perform across different inputs and how much effort is required to circumvent them.
2. Domain expert estimates, collected through surveys and/or structured forecasting exercises, of the

distribution of potential adversaries with varying capabilities and motivations, including their resourcing and willingness to invest time in circumvention attempts.

7.4 Documenting Mitigation Assessments

Some developers have begun to explore structured documentation approaches for their mitigation assessments. This approach is inspired by safety case approaches in other industries and may involve documenting key claims about system safety, supporting evidence, critical assumptions, and linking specific evidence from evaluations to safety claims. For example, Anthropic published a report explaining the design of the mitigation measures applied to the Claude 4 model, the rationale for the residual risk of the deployment being sufficiently low, and a set of ongoing operating requirements for the mitigations to maintain their effectiveness. This approach could also be used for autonomy-related risks. Like the documentation of frontier capability assessments, this method can facilitate internal decision-making and more transparent communication with external stakeholders about safety considerations.

Continuing Work

REQUEST FOR COMMENT

We welcome engagement with this and forthcoming technical reports from across the frontier AI safety and security ecosystem. Researchers and organizations interested in further refining and harmonizing the implementation of safety frameworks are invited to reach out to the Frontier Model Forum.

Please offer feedback at: info@frontiermodelforum.org

This report outlines current approaches to frontier mitigations, but as capabilities advance and our understanding of effective safeguards improves, developer approaches will be updated accordingly. Areas of continued work include:

- **Improving Mitigation Design and Implementation.** Developing more robust safety mechanisms that resist circumvention attempts; exploring new approaches to capability limitation and behavioral alignment that remain effective as models scale; and, establishing best practices for implementing controls across different deployment contexts – from API services to open-weight releases.

- **Advancing Assessment and Validation Methods.** Developing standardized approaches for testing individual controls and integrated systems; creating evaluation methods that remain effective as models acquire new capabilities like deception; and, establishing empirical validation of mitigation effectiveness against realistic threats.

- **Strengthening Security and Resilience.** Developing infrastructure security measures proportional to threat actors' or models' capabilities; creating safety mechanisms that persist despite modification attempts; and, building assessment frameworks that account for how threats might manifest through multiple attack vectors simultaneously.

Future research should focus on developing evidence-based approaches for the continuous improvement of mitigation strategies, helping them remain responsive to both emerging model capabilities and evolving threat landscapes as frontier AI systems continue to advance.