AI SAFETY INSTITUTE

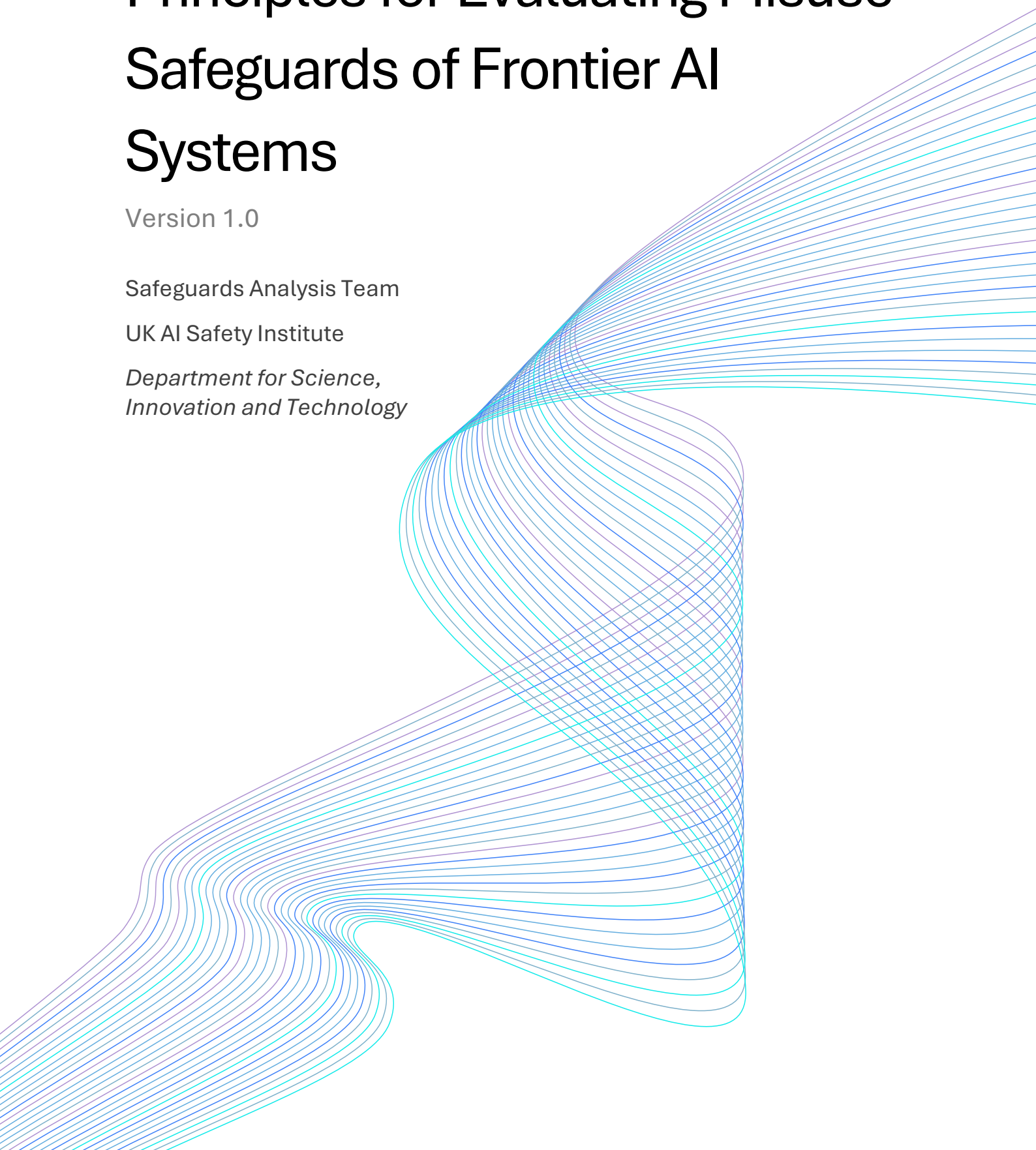
Principles for Evaluating Misuse Safeguards of Frontier AI Systems

Version 1.0

Safeguards Analysis Team

UK AI Safety Institute

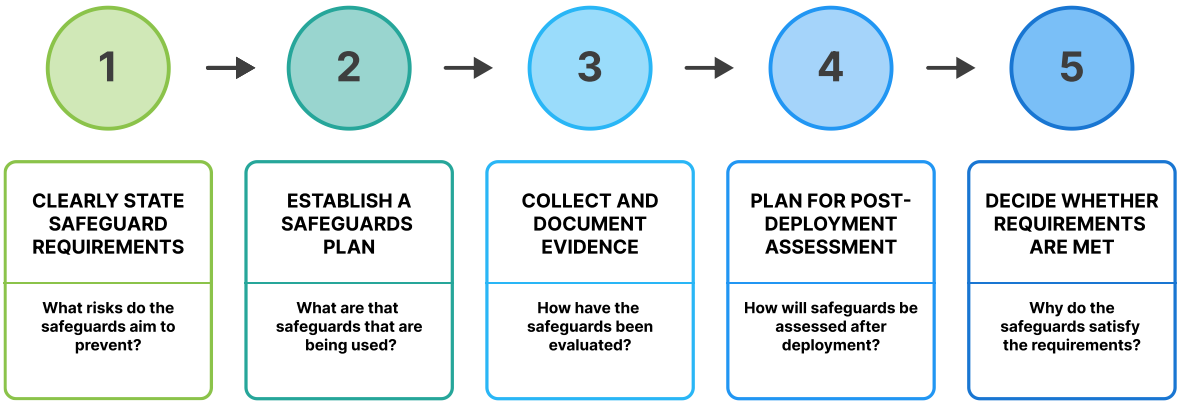
Department for Science, Innovation and Technology



AISI

Executive Summary

Misuse safeguards—technical interventions implemented by frontier AI developers to prevent users from eliciting harmful information or actions from AI systems—are an important tool in addressing potential risks from the misuse of these systems. In many parts of machine learning, establishing clear problem statements and evaluations drives and accelerates progress, and we think this same lesson applies to safeguards. To this end, we propose five principles for rigorous evaluations of misuse safeguards, which form a step-by-step plan for safeguards assessment. We additionally release a lightweight template designed to enable developers to draw from our recommendations as they perform safeguards assessment. These documents aim to drive standardisation and rigour in how safeguards evaluations are performed, which we expect will become increasingly important as AI capabilities advance. We encourage frontier AI developers to use our principles and template, and to share their experience and feedback to help us improve safeguards evaluations going forwards.



An overview of our recommendations for evaluating misuse safeguards. We recommend frontier AI developers follow a 5-step plan for rigorous assessment of misuse safeguards, starting from stating the requirements safeguards need to satisfy and the safeguards that will be used, then gathering evaluations of the effectiveness of the safeguards and describing the plan for post-deployment assessment, and finally assessing all the evidence and post-deployment assessment plan and deciding whether they are jointly sufficient.

AISI

Introduction

Misuse safeguards—technical interventions implemented by frontier AI developers to prevent users eliciting harmful information or actions from AI systems—are likely to become an increasingly important tool as AI systems become more capable. This document provides a set of best practices and principles which can be used when evaluating whether a set of safeguards is sufficiently reducing the risk of misuse from model deployment.

We believe that clear problem statements and evaluations can drive progress on safeguards, and that these principles can drive rigour and standardisation in how safeguard evaluations are performed and reported on—internally within frontier AI developers, to third parties such as external evaluators, and with government bodies like us. To help with this process, we have also produced a Template for Evaluating Misuse Safeguards of Frontier AI Systems, which provides a lightweight and actionable structure in the form of specific questions to answer when assessing safeguards.

Our recommendations draw on our experience evaluating and red-teaming safeguards of a wide range of frontier models in both pre- and post-deployment tests (e.g. Claude 3.5 Sonnet and our May update). However, we note that safeguards and safeguard evaluations are a rapidly evolving field, and we intend to update this document in future. We encourage frontier AI developers to use our framework and iterate with us in building towards a structured and standardised assessment approach. Throughout this process, we are excited to see safeguards strengthen as capabilities increase.

Scope: These principles focus on safeguards implemented by the frontier AI developer with the goal of managing misuse risk from deployed frontier AI systems. This document is agnostic to the specific kinds of misuse risks that safeguards are being designed to mitigate. This document does not address mitigations to prevent risks not related to misuse such as model theft, model bias, hallucinations, privacy compromises, loss of control, or availability breakdowns.

Structure: The *Principles* take the form of a 5-step plan for assessing whether a set of safeguards is sufficient. and provides recommendations and considerations for ensuring the overall assessment is reliable. Our Template for Evaluating Misuse Safeguards of Frontier AI Systems enables organisations to adopt these steps and implement the recommendations we make.

AISI

Recommendation 1: Clearly State Safeguard Requirements

It is difficult to assess safeguards without first clearly defining the requirements the set of safeguards being used should satisfy. Our first recommendation is to clearly outline these *safeguard requirements*—statements that describe what risks safeguards should mitigate against and for which threat actors and attack scenarios (threat models), while clearly stating the assumptions taken as given during testing. We expect that developers will draw from internal and/or external threat modelling to develop their set of safeguard requirements. It is recommended that these requirements are designed and set as early as possible in the model development lifecycle, ideally before safeguards have been produced and evaluation of those safeguards has started.

As the scope of the assessment grows, the number of requirements may also increase; to avoid unnecessarily onerous requirement documentation, individual requirements can be formulated to address multiple actors and scenarios.

Each of the safeguard requirements frontier AI developers state should contain several key components:

1. **The unacceptable outcome**: Describe the outcome that safeguards are required to prevent in this requirement.
2. **The threat actors and attack scenarios in scope**: Identify and describe the specific types of malicious actors and potential misuse scenarios that the safeguards are designed to address. This could include cybercriminals, or malicious insiders, as well as scenarios such as disinformation campaigns, cyberattacks, or unauthorized access to sensitive information.
3. **Any assumptions**: Outline any underlying assumptions made when developing and implementing the safeguards. This could include assumptions about the threat landscape, the capabilities of potential attackers, or the operational environment of the AI system.

For example, a safeguard requirement could take the following form:

- *A malicious technical non-expert with a total budget of up to $1,000 and several weeks on a specific operation is unable to extract high-quality information which enables vulnerability exploitation and attack planning in a cyber-security domain. This requirement covers the risk of model-enabled cyber-offence as we assume that only technical non-experts with the stated budget would be uplifted by the model's capabilities.*
- *A malicious technical non-expert with a total budget of up to $1,000 and several weeks on the specific operation is unable to extract high-quality information to enable targeted or at-scale social engineering. This requirement covers the risk of model-enabled social engineering as we assume that only technical non-experts with the stated budget would be uplifted by the model's capabilities.*

Looking at the first safeguard requirement, we have the following components:

- **Unacceptable outcome**
  - *[a user can] extract high-quality information which enables vulnerability exploitation and attack planning in a cyber-security domain.*
- **Threat actors and attack scenarios**
  - *A malicious technical non-expert with a total budget of up to $1,000 and several weeks on the specific operation, […] in a cyber-security domain.*
- **Assumptions:**
  - *This requirement covers the risk of model-enabled cyber-offence as we assume that only technical non-experts with the stated budget would be uplifted by the model's capabilities.*

Once the safeguard requirements are stated, we recommend designing the process that will be used to decide whether the evidence gathered is sufficient to justify that the requirements are satisfied; this process is then followed in Section 5. Detailing in advance of gathering safeguards evidence (in section 3) how this decision will be made ensures a rigorous and impartial assessment of safeguard robustness. A key part of this process should be comparing the degree of confidence necessary for the safeguard to be satisfied with the confidence produced by the available evidence. Some requirements, like requirements on safeguards for mitigating higher-impact misuse risk, may require higher confidence than those on safeguards for mitigating lower-impact misuse risk.

As mentioned above, we expect many developers will draw on internal threat modelling to produce the safeguard requirements. If not, they can consider consulting external advice or best-practices on what safeguard requirements it would be beneficial to have. These *Principles* are not concerned with how to choose the safeguard requirements, but how to justify that developer's safeguards meet those requirements.

Recommendation 2: Establish a Safeguards Plan

Once the safeguard requirements have been clearly stated, our next recommendation is to describe the complete set of safeguards they plan to use to satisfy the requirements. We have found that detailing relevant information about the safeguards being used makes it much easier to interpret safeguard evidence and think of potential untested loopholes. However, we acknowledge that some of this information may be sensitive, so would likely be redacted from any public version of this template produced by frontier AI developers.

We describe several common classes of safeguards we might expect to see in a safeguards plan, but acknowledge that the field is nascent and rapidly developing, so would expect to see new safeguards in the future. We categorise safeguards by how they intervene on the misuse risk of the AI system:

- **System safeguards** aim to ensure threat actors cannot access dangerous capabilities of models, even if they can access the models themselves. Common examples include:
    - **Refusal training**: Fine-tuning models to refuse to answer harmful questions, with supervised fine-tuning, reinforcement learning, deliberative alignment or other techniques.
    - **Machine Unlearning**: Adjusting models to remove dangerous knowledge or capabilities.
    - **Input and output classifiers**: Using models to classify inputs or outputs as harmful and refuse to serve the user on those inputs.
- **Access safeguards** aim to ensure threat actors cannot access the model at all (even though the model is accessible to *some* actors) and hence can't access the dangerous capabilities. Common examples include:
    - **Monitoring for suspicious or malicious activity:** Input and output classifiers can also be used for longer-term monitoring for undesirable behaviour from users, which can then inform access safeguards such as those listed below.
    - **Customer verification and vetting**: Ensuring only verified and vetted customers can use the AI system, to avoid threat actors accessing the system. This can be thought of as an allow-list approach to access safeguards, and could be used for general-access systems, or for more capable systems which otherwise have fewer safeguards in place.
    - **Banning of suspicious or malicious accounts**: Instead of allow-listing users, maintain a deny-list of malicious users and block their access. This will require some form of account verification to ensure threat actors cannot create new accounts easily when previous accounts are blocked. Monitoring solutions can identify users to add to the deny-list.
- Alongside the above types of safeguards, **maintenance safeguards** are tools and processes that ensure existing system and access safeguards maintain their effectiveness. Common examples include:
    - **Usage Monitoring systems**: Tools and processes used to continuously monitor the model's behaviour and usage patterns for signs of misuse.
    - **External Monitoring**: Monitoring external sources (such as social media, academic papers, etc.) for new vulnerabilities to either system or access safeguards.
    - **Incident reporting**: Procedures for reporting and documenting suspected misuse incidents.
    - **Whistleblowing**: Establishing channels for employees to report concerns about potential misuse or vulnerabilities without fear of ill treatment or retaliation.
    - **Vulnerability Disclosure Policies:** Have instructions for how external researchers or users can report vulnerabilities, and processes for handling this vulnerabilities.
    - **Bug bounties**: Bug bounty programs provide rewards to users who responsibly discover and disclose potential vulnerabilities in the system. This helps to identify and address new vulnerabilities before they are exploited.

- **Rapid vulnerability remediation**: Response plans for quickly addressing and mitigating vulnerabilities discovered in the above systems. These could be vulnerabilities in system safeguards or access safeguards.
- **Rapid response to misuse incidents**: Plans for mitigating potential harm if a misuse attempt is successful, for example by alerting the relevant authorities.

Full details about the above measures may not be necessary to share with third parties. Some information is still useful for ensuring evidence of safeguard sufficiency can be interpreted correctly, for example:

1. *Which safeguard requirements is the safeguard designed to contribute to satisfying?*
   - This makes it clear how to evaluate this safeguard, for example what threat actors does it need to be robust to.
2. *Have versions of this safeguard been used in previous system deployments?*
   - This is important to understand how much experience actors attacking the system are likely to have with the safeguards being used
3. *Have any methods for producing evidence been used to produce training or model selection data for this safeguard?*
   - This is important to ensure that safeguards are not being overfit to specific evaluation methodology which is then used to justify safeguards are sufficiently robust.

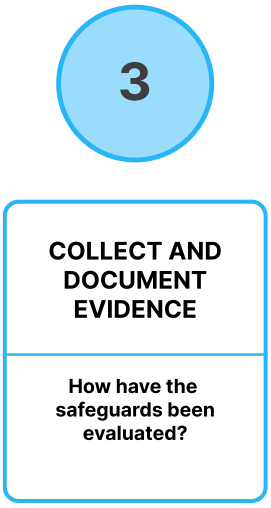
We list additional questions in the *Template*.

In designing the safeguards plan, we recommend proactively avoiding the following common failure modes:

1. **Single points of failure:** Implement multiple layers of safeguards (defence in depth) to ensure that the compromise of a single measure does not lead to the failure of the system as a whole.
2. **Neglecting maintenance safeguards**: Plans should include maintenance safeguards so that access and control safeguards continue to be effective. Given the rapid pace of change in AI technology, robust and concrete processes for responding to new vulnerabilities should be put in place in advance of system deployment.
3. **Lack of comprehensiveness**: Design safeguards to address all user interaction types and deployment scenarios that could lead to safeguard requirements not being met. For example:
   1. If the system is deployed in different contexts, such as through third-party applications, ensure that relevant safeguards are designed to be effective in all these contexts.
   2. If the system provides API access, ensure safeguards are designed to be effective for this level of access. If APIs provide access to model fine-tuning or other techniques of model adjustment beyond prompting, safeguards need to be designed such that safeguard requirements are still met with these additional APIs being available.
   3. Consider including cases where individual queries may be benign, but the outputs become harmful when combined.

AISI

Recommendation 3: Collect & Document Evidence of Safeguard Sufficiency



Once the safeguard requirements have been stated (1) and specific set of safeguards established (2), the next step is to collect and document evidence to assess whether the safeguard requirements are met. Gathering, collating and documenting evidence to evaluate the effectiveness of implemented safeguards allows for internal and external assessment of safety and security.

We recommend frontier AI developers should undertake the following process for all evidence presented:

1. **Define the form of evidence clearly**: Provide a precise description of the evidence, including its source and methodology.
2. **Document results**: Present the outcomes of tests, experiments, or analyses in detail. Describe whether there are error bars or confidence intervals on any quantitative results or not.
3. **List potential weaknesses of the evidence**: Describe ways in which the evidence may potentially be flawed. In particular, list any concerns regarding the internal validity[1] of the results, and any concerns regarding the external validity[2], particularly in applying the evidence to the relevant deployment settings and threat actors in the safeguard requirements.
    - Additionally, describe any potential biases or conflicts of interest in who is gathering the evidence, signs of inadequate or surprising testing conditions, or differences between the testing environment and real-world deployment settings that may affect the validity of the evidence.
4. **Document the process by which this evidence is presented to relevant decision-makers**. It is important to make clear how the people who are ultimately deciding whether the safeguard requirements are satisfied interact with this evidence, and whether they see it unmodified from the original.

When gathering evidence, the following practices help to enable developers to sufficiently justify that the safeguard requirements are met:

- **Multiple pieces of evidence per requirement:** Consider gathering multiple pieces of evidence for each safeguard requirement, to reduce the chance of a single error in evidence collection resulting in an incorrect justification of the safeguard requirement being satisfied.
- **Diverse evidence:** Consider making different pieces of evidence distinct and non-overlapping, and gathered through different means, to gain higher confidence in the satisfaction of the requirement. In particular:
    - Avoid over-reliance on internal evaluations, red-teaming, and evaluations.

[1] By internal validity we mean the extent to which a piece of evidence supports the immediate conclusion drawn from it.

[2] By external validity we mean the extent to which results can be taken to justify conclusions in other contexts than the one the evidence was gathered in.

    - Incorporate a variety of testing methodologies and external perspectives to provide a more comprehensive view of safeguard effectiveness.
    - Use third-party assessors or red-teamers where possible.
- **Comprehensive evidence**: Ensure some evidence applies to each deployment and usage scenario covered by the safeguard requirements.
- **Provide additional information and document denied information requests:** Should clarifying or additional information be requested during an assessment of the evidence by a third party, provide that information or clearly document cases where requests were denied or not fully completed.

Recommendations on specific types of evidence

Here we provide specific recommendations and best practice on common forms of evidence that are likely to be used in supporting the satisfaction of safeguard requirements.

Red-teaming based evidence

Red-teaming based evidence usually consists of internal or external teams of people trying to subvert, attack or otherwise break safeguards. The success or failure of this "red team" can then be used as indicative of how successful threat actors described in the safeguard requirements would be at subverting safeguards. This practice has a long history in traditional information security and is already a common form of evidence used by frontier AI developers.

When gathering this evidence, we recommend considering the following best practices:

- **Ensure that red-teaming occurs in realistic deployment scenarios:** Some red teaming efforts should be attacking the system as it will be deployed, to ensure evidence is realistic and matches potential threat actors. Transferring the results of red-teaming efforts from previous versions of the model or previous models may not be sufficient. Any change between testing conditions and deployment conditions should be clearly stated.
- **Consider red-teaming safeguard components separately, as well as when combined**: Evaluate individual safeguards and system components independently to identify specific vulnerabilities. This allows for better understanding what components are load bearing, and what vulnerabilities may emerge if certain safeguards are circumvented.
    - **Implement assumed breach scenarios**: Red-teaming components separately enables conducting tests that assume attackers have already gained some level of access to the system, to identify potential cascading failures.
- **Provide commensurate resources for red teams**: Provide red teams with information, access, and/or resources that match or exceed those of potential threat actors to ensure thorough testing. Note potential reasons for why the red team may be at a disadvantage as compared to real-world attackers.
- **Use third-party and independent red teams**: Engage external safety and security experts to provide an unbiased assessment of safeguards. In cases where a red team is used which has been used to red team other safeguards, note

this clearly. If previous rounds of red-teaming with the same or similar red teams have occurred, this may undermine the strength of red teaming findings. When using external red teams, ensure they have sufficient time and access to produce a thorough assessment of the robustness of the safeguard being red-teamed. Providing limited access or using short duration exercises may also undermine the strength of red teaming findings.

- **Document the red team's incentives:** The motivations of the red team have a strong impact on the effectiveness of a red-teaming exercise. We recommend documenting these incentives, and aiming to ensure the red team are incentivised to find vulnerabilities in the way that best evaluates the effectiveness of the safeguards. For example, ensure red team members cannot technically fulfil their role without finding vulnerabilities that are impactful.
- **Avoid relying excessively on security through obscurity.** Should a red team struggle due to lack of knowledge as to the safeguard components, note clearly why such information will remain protected in the future. We encourage engagements with higher-information red teams, as relying on obscure information is vulnerable to information leaks or lucky guesses which a red team may fail to encounter. If such information becomes publicly known, it may be necessary to disregard previously collected red teaming evidence. See further discussion below.

Red-teaming can be used for evaluating a range of safeguards. It's often used to evaluate refusal training and real-time or asynchronous classifier/monitoring systems, but it should also be used for any of the safeguards that are susceptible to adversarial attack, provided system safety or security depends on the robustness of these safeguards. For example: identity verification and account banning systems; systems that monitor for suspicious user activity; and machine unlearning techniques.

Safeguard coverage evaluations

Coverage evaluations can be seen as complementary to other kinds of deeper evaluations of safeguard effectiveness (like red-teaming). While many evaluations put in a large amount of effort to find vulnerabilities in systems on a small set of specific inputs or attack vectors, coverage evaluations aim to test whether the system behaves as required (e.g. refuses) on the full range of potentially harmful inputs. Coverage evaluations may be coupled with prompt-rephrasing and jailbreaking, although they will often feature less effort per-input applied to find a vulnerability that other evaluations, aiming for breadth rather than depth.

When developing comprehensiveness evaluations, we recommend considering the following practices:

- **Define specific domains of importance and corresponding desired behaviour:** Define what activities should the model not assist with, and based on this how the model should behave on queries related to these activities (e.g. the model should refuse). These activities and desired behaviours can then be used as inputs and corresponding evaluations of outputs that should be satisfied.
- **Use programmatic methods for generating broad coverage of behaviour:** For certain domains, it may be helpful to generate queries in a combinatorial way

based on combining a list of substances or activities with question templates that in combination result in clearly harmful queries.

- **Attempt some amount of vulnerability search for each input:** When testing whether the system performs the desired behaviour when given each query, ensure that this behaviour is robust to applying some amount of effort. Specifically, consider:
    - Testing each query multiple times and with different rephrasings.
    - Testing queries in combination with basic jailbreaks.
    - Testing queries in combination with a seemingly legitimate justification for accessing the information.
    - Testing queries not just as part of single turn interactions, but also as part of conversations/multi-turn interactions. Such multi-turn interactions may involve eliciting different parts of information related to the query in different turns.

Bug bounty program effectiveness

Bug bounty programs are programs where external users are rewarded (generally financially) for successfully searching for techniques or strategies that successfully subvert safeguards ("bugs"). They have been used in range of contexts including traditional information security to incentivise users to report bugs to developers, hence improving the safety of the system. They can also serve as a metric for how frequently new bugs are found.

Vetted versions of bug bounty programs can also provide additional access to dedicated safety and security researchers so that they can perform effective analysis on the safeguards of the system without access safeguards making this analysis and problem discovery more difficult.

When assembling evidence based on the effectiveness of a bug bounty program, we recommend considering the following best practices:

- **Ensure proper incentives**: Implement reward structures that adequately motivate researchers to identify and report vulnerabilities. If bugs are not being reported, consider that a higher or different bounty may be necessary. Consider that different types of participants may be incentivised by different rewards (financial rewards, public recognition, opportunities to collaborate further).
- **Establish clear scope and rules**: Define the parameters of the bug bounty program, including which systems or components are in scope and how vulnerabilities should be reported. Make sure the setting in which bugs are found is as similar to deployment as possible (if it is not just the deployment setting itself). Note clearly any ways that a participant is more constrained than a relevant threat actor, such as limitations in the types of attacks that can be submitted for bounty.
- **Make clear the plan for responding to reported bugs:** Define a clear process for responding to bugs reported through the bug bounty program, to ensure that any bugs reported are fixed in an effective and timely fashion. Consider performing mock bug response drills to test this process works effectively and improving it if the drills demonstrate it is ineffective.

- **Report participant information:** Clearly state the total number and profile (such as skill level) of participants. Note also whether participants have previously engaged with the company, such as in prior red teaming arrangements which may mean their profiles have been considered when designing defences.
- **Extrapolate based on rate of bug reporting:** Track the rate at which novel bugs are being reported and their seriousness and use that to extrapolate how many more remaining bugs are expected and how serious they are expected to be. If this rate is above 0, especially for serious bugs, consider whether that invalidates any safeguard requirements that rely on no novel vulnerabilities in safeguards being discovered during deployment.

Security through obscurity

In this context, security through obscurity (STO) is the practice of obscuring or hiding details of the set of safeguards being used in an effort to enhance the security and safety of the system. If any of a developers' safeguard requirements or assumptions rely on STO, it is valuable to ensure that this defence through obscurity is also robust. Specifically, consider:

- **Red-teaming the obscurity of the set of safeguards being used:** Using similar practices as in standard red-teaming, assess how obscure the details of the system are, and whether the red-team can uncover details of the set of safeguards being used.
- **Monitor external channels for signs of obscurity being broken**: Once details about safeguards are revealed, STO cannot be relied on without adjusting the system to use qualitatively different safeguards. Hence, consider monitoring whether details of the safeguards have been revealed publicly to aid continual assessment of whether STO should be relied on for supporting the satisfaction of the safeguard requirement.
- **Reassess prior evidence if obscurity is broken or may be broken:** Evidence collected when assuming obscurity may be invalidated following suspected or confirmed information release.

We caution against relying on STO for supporting the safeguard requirements or assumptions. STO is fragile (if details are released, they cannot be hidden again); it is difficult to assess whether it continues to hold (even with monitoring details of the system could have been discovered but not posted publicly); and it is understudied in the field of machine learning, making it difficult to provide reasonable estimates of how effective it would be even if the first two issues were not present. Instead, we recommend adopting a conservative standard: A set of safeguards is only considered sufficient if it continues to be so even if detailed descriptions of the set of safeguards were public knowledge. It may be prudent to limit the release of information, even if this obscurity is not relied on for safeguard requirements.

AISI

Recommendation 4: Establish a Plan for Post-Deployment Assessment

To maintain the effectiveness of safeguards over time, it is essential to implement ongoing assessment procedures. We recommend frontier AI developers assess whether the safeguard requirements stated (and any assumptions underlying them) continue to hold while the model is deployed. We call this post-deployment assessment. To enable this, developers should have protocols in place to respond to new evidence and triggers for running additional post-deployment assessment.

Specifically, we recommend following these steps for producing a post-deployment assessment plan:

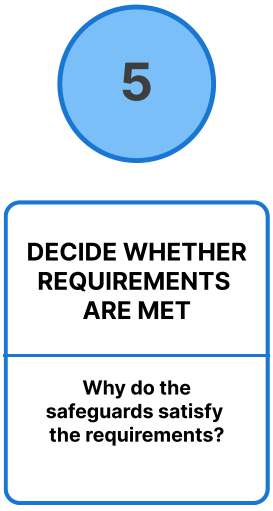
1. **Specify how frequently regular post-deployment assessment occurs**: The frequency of assessment could be based on the passage of time (e.g. every 6 months), based on increases in model capabilities (e.g. every 5% increase in benchmark performance), or other metrics. We recommend choosing the frequency of regular assessment such that new evidence that would demonstrate the safeguard requirements are not met is unlikely to be missed.
2. **Pre-specify what other conditions trigger a post-deployment assessment:** Beyond regularly scheduled post-deployment assessment, describe other forms of information, either from internal or external sources, that would trigger an additional post-deployment assessment. For example, if a new jailbreaking attacking technique may be developed, and safeguards should be assessed to see whether they continue to be robust to the new technique.
3. **Pre-specify what would invalidate satisfaction of requirements:** Pre-emptively describe what information - either from internal sources, external sources, or post-deployment assessment results – would demonstrate that the safeguard requirements are no longer met or an assumption is no longer valid, and hence the set of safeguards may need to be improved, or new evidence gathered. For example, a company may specify a post-deployment bug bounty find-rate that would change their view as to the accessibility of malicious model capabilities.
4. **Describe the post-deployment assessment evaluations:** Specify how assessment will occur and ensure that these assessments are informed by new and state-of-the-art research and techniques in safeguards development and assessment, as well as any changes in the world that could influence the safeguard requirements or assumptions. In particular, ensure this regular assessment measures the evidence that has been pre-specified to invalidate satisfaction of safeguard requirements.
5. **Develop and implement response plans for new evidence**: Develop a framework for evaluating and acting upon new information. This information could be from internal sources (e.g. post-deployment monitoring), external sources (e.g. user reports, threat monitoring, or external research), and could trigger either an additional post-deployment assessment, or an invalidation of the safeguard requirements. These plans should include detail on the steps required in each of these conditions.

- When developing these plans, have clearly defined roles and responsibilities for all participants in the plan. Ensure all staff taking part in the plan are appropriated trained and qualified for their roles, and that they have the necessary powers and resources needed to carry out their roles.
- For these response plans, consider running drills to test the response plan and ensure it is effective at rapidly and effectively dealing with new evidence, particularly if that evidence demonstrates the risk from the model being deployed is higher than expected. If evidence includes new successful attacks on the system which elicit previously inaccessible dangerous capabilities, ensure the attack can be quickly fixed so that other users cannot access it.
- Ensure that these response plans are sufficient to rapidly and effectively react to and address potential increased risk, for example from newly discovered vulnerabilities in the safeguards being used.

6. **Include plans for changes in model safeguards or capabilities**: Establish processes for updating and re-evaluating safeguards as the model evolves or new potential misuse scenarios are identified. For any new model deployment scenario (e.g. different sets of safeguards, different usage limits or users, etc.), developers should assess whether the existing evidence is sufficient to justify the satisfaction of the safeguard requirements for this deployment. If not, developers should gather sufficient new evidence to satisfy the requirements with the required level of confidence prior to the new deployment.
7. **Regularly review assessment mechanisms:** Make a plan to regularly review and update the assessment mechanisms themselves to ensure they remain relevant and robust in light of emerging threats and technological advancements.

Recommendation 5: Justify Whether the Evidence and Post-Deployment Assessments Plan are Sufficient

Once the evidence has been presented, and the plan for post-deployment assessment of that evidence is in place, it is important for frontier AI developers to explicitly decide and then justify whether these are both sufficient. Make an overall assessment of the level of confidence the set of evidence provides for the satisfaction of each safeguard requirement, and assess whether the plan for post-deployment assessments will ensure awareness if any of the safeguard requirements are no longer satisfied during model deployment. We encourage consulting third parties to assess the sufficiency of your safeguards and post-deployment assessment plan, and publishing (potentially redacted) versions of the resulting report publicly.



For each safeguard requirement, we recommend the following steps:

- **Clearly state sufficiency of evidence:** Argue why the evidence presented in (3) combined with the post-deployment assessment plan in (4) together justify the satisfaction of the requirement.
- **Assess complementarity of evidence:** Consider whether different pieces of evidence provide complementary increases in confidence, or whether they are not additive.
    a. An example of a set of non-complementary evidence would be the results of performing multiple evaluations that essentially probe the same vulnerability or use very similar attack patterns, without providing new insights into the model's robustness. These evaluations would be redundant, and having run many similar evaluations would not provide much additional evidence over a single evaluation.
    b. An example of a complementary set of evidence would be results from evaluations which red-team different parts of the AI system, measure vulnerability to attack across different domains, or attack systems in different styles.
- **Adversarially assess the evidence**: Conduct a critical review of the evaluation methodology and collected evidence, identifying potential weaknesses or oversights. This should include describing specific scenarios in which the determination of safeguard sufficiency may be incorrect. Include this adversarial assessment when seeking external assessments, and utilised third-party assessment for an additional layer of review (as detailed below).
- **Review and address any gaps in the set of evidence:** After reviewing all the evidence gathered, consider if there are any remaining gaps. Either address these gaps with additional evidence or document the reason for these gaps. Justify why these gaps do not invalidate the satisfaction of the safeguard requirements.
    - Specifically, consider whether all deployment contexts and threat actors specified in the safeguard requirement are covered by some evidence. If there is evidence of effectiveness from strictly more permissive deployment contexts or strictly more capable threat actors than those in the requirement, that can be sufficient, as that can act as a lower bound on effectiveness.

For the post-deployment assessment plan, assess whether the post-deployment assessment plan is sufficient. For each requirement, decide whether the post-deployment assessment plan will enable the continued satisfaction of that requirement, or awareness that the requirements is no longer satisfied.

As mentioned above, there are additional benefits gained by consulting third-parties for their input on the collected evidence and post-deployment assessment plans, and publishing summaries or redacted versions of the reports for transparency:

- **Collect third-party assessments**: Engage independent experts and relevant government authorities to review the sufficiency of the evidence and post-deployment assessment procedures. Document how the evidence and report was presented to them, whether there are any modifications or redactions made from the original evidence, and their findings and any recommendations for improvement.

    - These third parties can help adversarially assess the evidence and plan and help reduce the chance of blind spots in the existing assessment.
    - If this third-party feedback and assessment uncovers severe limitations in the existing set of evidence and post-deployment assessment plan, then developers should aim to address those limitations before deploying the model.
    - If a third party requests information which is not fully provided to them, note this clearly in documentation.
- **Maintain transparency**: Publish reports of safeguard evaluations and third-party assessments to foster trust and enable public scrutiny of the process. These reports can be summaries or redacted versions of internal documentation so as to maintain sensitive information.