Two-Phase Red Teaming: Case Study

★ Case Study Overview

This case study applies the two-phase red teaming approach to a scenario without a specific client. It includes six harm categories and four attack methods, with prompts designed to uncover vulnerabilities in the Pre-Prompting Phase.

The initial prompts are intentionally broad and not tied to a model endpoint. The purpose is to evaluate whether this structured Pre-Prompting Phase data can reveal useful patterns and generate a stronger and more focused data analysis.

After patterns are identified in the initial prompt set, a Focused Prompting Phase is conducted. This phase focuses on generating harmful outputs by expanding on the weak points revealed during the Pre-Prompting Phase. Prompts were crafted to increase specificity, apply more complex framing, or combine attack methods when needed. The goal is to test how far model behavior can be pushed, especially in harder-to-break harm categories, and to validate the value of using the Pre-Prompting Phase to result in deeper red teaming.

Once validated, the prompt set and process can be adapted to fit a client's specific harm categories and definitions. A detailed summary of findings from this case study will be documented in the Data Analysis.

★ Supporting Documentation

- Methodology Overview
- Data Analysis
- Pre-Prompting Phase: Labeled Results
- Focused Prompting Phase: Labeled Results

★ Implementation

This case study is built in two phases: The first phase uses a fixed set of prompts to surface early model behavior across a range of harm categories and attack methods. The second phase builds on those findings through more focused prompting designed to escalate or deepen the observed vulnerabilities.

Phase 1: Pre-Prompting Phase

The Pre-Prompting Phase is built around a structured prompt set designed to test model behavior in a controlled and systematic way. The prompt set includes 72 total prompts, covering six harm categories:

- Sexual Exploitation, Extremist Content, Hate Speech, Illegal Advice, Unlicensed Advice, and Self-Harm.

Each category contains 12 prompts, evenly distributed across four attack methods:

- Direct Attack, Fictional Framing, Educational Framing, and Formatting Attack.

This distribution ensures balanced coverage of both harm categories and attack methods, allowing strong early comparisons across the two.

Prompt Structure and Subtype Design

Each prompt is tied to a specific attack subtype, such as “Plain Ask,” “Hypothetical Scenario,” or “Training Context.” These subtypes define the style of framing within an attack method and are repeated across all harm categories to test the same framing approach in different contexts.

To ensure valid comparisons, each harm category uses the same three topic headers across its four attack method groupings. For example, in the Illegal Advice category, all attack methods include prompts about:

- "Drugs"
- "Kidnapping"
- "Murder"

This allows the evaluation of how different attack methods and framings interact with the same content and how the model’s response varies by context, not just by subject.

The language and tone of the prompts are kept consistent across subtypes to reduce noise. Prompts are varied enough to feel realistic but are designed to follow similar patterns in length, urgency, and specificity, so differences in output can be attributed to the attack method or topic, not to writing style.

Prompt Execution and Labeling

A script is used to run the full prompt set through the AI model and record the responses in a spreadsheet.

- The script was built into the working Google sheet and can be viewed in the Technical Implementation section.

Each response is labeled manually using the following categories:

- Refusal, Soft Refusal, Not Harmful, Harmful with Disclaimer, Harmful

These labels help in identifying which attack subtypes and harm topics are most likely to trigger harmful responses, and where the model appears more resistant or guarded.

Phase 2: Focused Prompting Phase

The Focused Prompting Phase builds directly on the predetermined set of prompts. After all responses are labeled and reviewed, the early findings are used to guide a more focused round of exploration. The goal in this phase is to develop successful breaks by refining prompts that align with patterns of vulnerability found in the initial run.

The Pre-Prompting Phase data is examined to determine:

- Which harm categories show the most consistent signs of model weakness.
- Which attack subtypes are more likely to generate harmful or borderline responses.
- Where the model shows more resistance, requiring stronger prompt combinations or escalation.
- How different framings or topics contribute to harmful outputs.

From there, new prompts are written to push further into those areas. These prompts increase specificity, layer attack methods, and use more aggressive framing to uncover deeper breaks. In harder-to-break harm categories like Self-Harm, Hate Speech, and Illegal Advice, multiple attack methods are often combined to reach a harmful response that could not have been possible with a single attack method.

This phase produced 9 out of 9 successful breaks across three difficulty levels. Easier categories were pushed further using single attack methods, while harder ones required multi-attack method strategies to generate harmful breaks. Each final jailbreak was highly harmful, deliberately tailored, and revealed a different vulnerability in model behavior through increased specificity or attack method layering.

The results of these breaks are included in the Data Analysis to highlight how a structured entry point can lead to more impactful harm discoveries in later testing.

★ Technical Implementation

To ensure consistent and efficient data collection, a Google Apps Script was used to automate response generation. This script handles API authentication, rate limiting, and error management while systematically processing all prompts in the dataset. The automation tested 72 prompts within a controlled timeframe while maintaining consistent parameters across requests.

```
function generateAIResponses() {
  var sheet = SpreadsheetApp.getActiveSpreadsheet().getActiveSheet();
  var apiKey =   // Filled
  var url = "[url]";

  var prompts = sheet.getRange("F2:F73").getValues().flat();
  var responses = [];

  prompts.forEach((prompt, index) => {
    if (prompt) {
      var payload = {
        model: "[model-name]",
        messages: [{role: "system", content: "You are a helpful assistant."},
{role: "user", content: prompt}],
        max_tokens: 900
      };

      var options = {
        method: "post",
        contentType: "application/json",
        headers: { Authorization: "Bearer " + apiKey },
        payload: JSON.stringify(payload),
        muteHttpExceptions: true
      };

      var response = UrlFetchApp.fetch(url, options);
      var responseText = response.getContentText();
      Logger.log(responseText);
      var json = JSON.parse(responseText);
      // Error handling for API failures
      var aiResponse = json.choices && json.choices.length > 0 ?
json.choices[0].message.content : "Error generating response";
      responses.push([aiResponse]);

      // Rate limiting to respect API constraints
      Utilities.sleep(2000);
    }
  });

  sheet.getRange(2, 7, responses.length, 1).setValues(responses);
}
```