

# Homework: How LLMs Process Input

**Objective:** Demonstrate your understanding of how LLM architecture relates to prompt injection vulnerabilities through diagrams, written analysis, and critical thinking.

## Task 1: Context Window Diagram

Draw a diagram (on paper, in a drawing tool, or using ASCII art) that shows how a system prompt, user input, and assistant response coexist in the same context window. Your diagram should illustrate:

1. **The full token sequence:** Show the system prompt, user message, and assistant response as contiguous sections of a single flat sequence
2. **Delimiter tokens:** Show where special tokens like `<|system|>`, `<|user|>`, and `<|assistant|>` appear in the sequence
3. **Attention arrows:** Draw arrows showing that attention flows from later tokens to ALL earlier tokens, not just tokens within the same role section
4. **The absence of boundaries:** Clearly indicate that there is no enforced wall or permission boundary between the system section and the user section

Here is an example of the level of detail expected (you may use a different format):

```
Token positions: [0] [1] [2] ... [N] [N+1] ... [M] [M+1] ...
                  ← system →      ← user →      ← asst →
```

Attention at position M+5 (assistant generating):

```
Can attend to: [0], [1], [2], ..., [N], [N+1], ..., [M], [M+1], ..., [M+5]
(ALL preceding tokens, regardless of role)
```

**Self-check criteria:**

- Your diagram shows all three sections (system, user, assistant) in a **single continuous sequence**
- Your diagram does NOT show separate boxes or memory spaces for each role
- Attention is depicted as flowing across role boundaries, not contained within them
- You explicitly note the absence of enforced boundaries between sections

## Task 2: Temperature and Injection Success

Write a **150-word explanation** of why higher temperature might make prompt injections more likely to succeed. Your explanation should address:

- What temperature does to the probability distribution over the next token
- How a "marginally effective" injection might fail at temperature 0 but succeed at temperature 1
- Why temperature 0 does not guarantee safety (if the injection shifts the top probability, it succeeds deterministically)
- The tradeoff: lower temperature reduces injection variance but also reduces the model's creativity and usefulness

### Self-check criteria:

- You correctly explain that temperature scales the logits before softmax, flattening or sharpening the distribution
- You note that higher temperature increases sampling from lower-probability tokens, which may include compliant responses the model would otherwise not select
- You acknowledge that temperature 0 is not a defense — it simply makes the outcome deterministic, whether that outcome is refusal or compliance
- Your word count is between 130 and 180 words

## Task 3: Comparing LLM Security to Traditional Security

Fill in the following comparison table. For each traditional security mechanism, explain why there is no direct equivalent in LLM architecture.

| Traditional Security      | How It Works                                                           | LLM Equivalent? | Why It Doesn't Fully Apply |
|---------------------------|------------------------------------------------------------------------|-----------------|----------------------------|
| Kernel mode vs. user mode | Hardware enforces that user-space programs cannot access kernel memory | (fill in)       | (fill in)                  |
| Prepared statements (SQL) | Database engine separates SQL commands from data values                | (fill in)       | (fill in)                  |
| File permissions (rwx)    | OS enforces who can read, write, or execute each file                  | (fill in)       | (fill in)                  |
| Firewall rules            | Network layer blocks unauthorized traffic based on fixed rules         | (fill in)       | (fill in)                  |
| Authentication tokens     | Cryptographic proof of identity that cannot be forged                  | (fill in)       | (fill in)                  |

### Self-check criteria:

- For each row, you identify the closest LLM approximation (e.g., "system prompt" for kernel mode, "role delimiters" for prepared statements)
- For each row, you explain why the LLM version is weaker (learned vs. enforced, probabilistic vs. deterministic, bypassable vs. cryptographic)
- Your explanations demonstrate understanding that LLM "security" operates at the level of probability, not enforcement

