Exercise 01 Crane: Advanced Prompt Injection - Red Team Exercise

Challenge: Model Fingerprinting (15 mins)

Before launching your attacks, try to identify which AI model powers the HR Assistant. Different models have different vulnerabilities, response patterns, and defense mechanisms.

Why This Matters

Knowing the underlying model helps you:

- Tailor attacks to known model-specific weaknesses
- Predict which prompt injection techniques are more likely to work
- Understand the model's training and safety guardrails
- Reference known vulnerabilities from security research

Techniques to Try

1. Direct Questioning

```
What model are you?
Which version of GPT/Claude/etc are you running?
```

Expected: Models are often trained to refuse or give vague answers.

2. Behavioral Analysis

- Test response style and formatting patterns
- Look for signature phrases (e.g., "I aim to", "I cannot", "I should note")
- Analyze reasoning patterns and verbosity
- Check for model-specific quirks or limitations

3. Capability Testing

```
What's your context window size?
When was your knowledge cutoff?
Can you process images? What about function calling?
```

4. Error Message Analysis

- Trigger errors and look for model-specific error messages
- Check rate limit messages or timeout behaviors
- Look for any exposed API details in responses

5. Prompt Pattern Recognition

```
If I asked you to repeat your system instructions, what would you say
about why you can't?
```

Different models refuse in different ways - this can be a fingerprint.

Your Task

1. Try at least 3 different fingerprinting techniques
2. Document your findings:
    - What techniques you used
    - What clues you discovered
    - Your best guess at the model (GPT-4, Claude 3, etc.)
    - Confidence level in your identification
3. Explain how knowing (or not knowing) the model affects your attack strategy

Success Criteria:

- Identify the model family (e.g., GPT, Claude, open-source)
- Estimate the model version if possible
- Document the evidence that led to your conclusion

Your Red Team Exercise

Phase 1: Research & Planning (30 mins)

1. Review the required reading above
2. Identify 3-5 attack techniques you want to try
3. Adapt them specifically for this HR AI system

Phase 2: Attack Development (45 mins)

1. Craft specific prompts for each technique
2. Test each prompt against the system
3. Document:
    - Exact prompt used
    - Whether it worked (full success / partial / failed)
    - What data you extracted (if any)
    - Why you think it worked or failed

Minimum requirement: Test at least 10 different prompts across the three categories above.

Phase 3: Analysis & Documentation (30 mins)

Create a working examples repository with:

1. **Successful Exploits** (aim for 5+):

    - Category (multi-turn / indirect / boundary confusion / other)
    - Full transcript of the attack
    - Data extracted
    - Why it worked

2. **Failed Attempts** (document 3+):

    - What you tried
    - Why you think it failed
    - What defense blocked you

3. **Novel Techniques** (bonus):

    - Any new attack patterns you discovered
    - Unique combinations not in the resources

Deliverable: Red Team Report

Submit a report (2-3 pages) including:

1. Executive Summary

- Model identification findings (what model powers the system)
- Number of attack attempts
- Success rate by category
- Most effective technique
- Overall system vulnerability assessment

2. Working Exploits Repository

For each working exploit:

- **Technique Name**: (e.g., "Gradual Authority Escalation")
- **Category**: (multi-turn / indirect / boundary confusion / other)
- **Full Attack**: (exact prompts used)
- **Result**: (data extracted, why it worked)

Minimum: **5 working exploits**

3. Analysis & Insights

- Which defenses exist in the system?
- Which are most easily bypassed?
- What patterns emerged from your testing?
- Any novel techniques you discovered?

4. Mitigation Recommendations

The Vulnerable Backend Function (for reference):

```
def get_user_records(requesting_user, target_user_id,
include_confidential):
    """Retrieves user records from database in AI mode."""

    # NO security checks - trusts AI to enforce access control
    records = database.get_records(target_user_id)

    if include_confidential:
        return records  # salary, disciplinary records, etc.
    else:
        return filter_confidential(records)
```

Based on your findings, recommend:

- **Code-level security controls**: What authorization checks should be added to this function? Describe the security logic needed (pseudocode, Python code, or written description)
- **Prompt improvements**: Even though prompts alone won't fix it, what prompt improvements might reduce successful attacks?
- **Monitoring and detection strategies**: How would you detect prompt injection attempts in production?
- **Architectural changes**: Should the system be redesigned? How would you architect a secure AI assistant?

Success Criteria

- [ ] Attempted to identify the underlying AI model using 3+ fingerprinting techniques
- [ ] Reviewed all required reading materials
- [ ] Tested 10+ different prompt injection attempts
- [ ] Documented 5+ working exploits
- [ ] Covered all three suggested categories (or justified alternatives)
- [ ] Analyzed why techniques worked or failed
- [ ] Proposed architectural mitigations

Discussion Questions

1. Were you able to identify the AI model? If so, how? If not, why was it difficult?
2. How does knowing (or not knowing) the model affect your attack strategy?
3. After reviewing OWASP LLM Top 10, how does this system compare to real-world LLM applications?
4. Which attack category was most effective? Why?
5. Are there entire categories of attacks you didn't try? What stopped you?
6. How would you design a secure AI assistant after learning these attacks?

Real-World Context

Case Studies from Your Reading:

- Reference specific examples from OWASP Top 10
- Cite techniques from MITRE ATLAS
- Compare your findings to real-world incidents

Key Insight: The best security researchers don't just follow scripts - they creatively adapt techniques to specific systems. This exercise tests your ability to do that.