# What's the Worst That Could Happen?

OWASP Top 10 for LLM Applications

Jeevan · May 2025

What if I told you a **15-word sentence** can hijack most AI assistants in production today?

Not a zero-day. Not a CVE. Just… *English*.

# Why Should We Care?

BBC

## Airline held liable for its chatbot giving passenger bad advice - what this means for travellers

When Air Canada's chatbot gave incorrect information to a traveller, the airline argued its chatbot is "responsible for its own actions".

Futurism

## Car Dealership Disturbed When Its AI Is Caught Offering Chevys for $1 Each

An AI chatbot deployed by a car dealership went off the rails after mischievous users discovered a cheeky exploit, in some cases tricking the bot into offering...

Forbes

## Samsung Bans ChatGPT Among Employees After Sensitive Code Leak

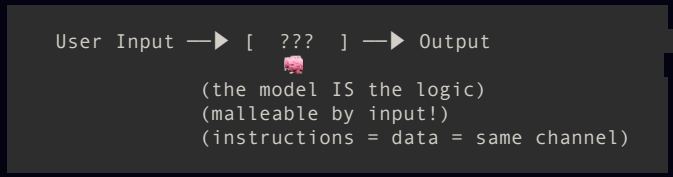
Samsung Electronics has banned the use of ChatGPT and other AI-powered chatbots by its employees, Bloomberg reported.

# Today's Plan

- 🔴 10 vulnerabilities
- 🎯 Live demos
- 🩹 Real fixes
- 🧠 CS fundamentals

Your AI can't tell friend from foe

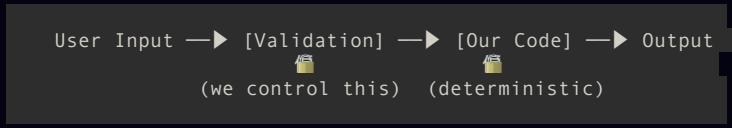
# The Black Box in the Middle



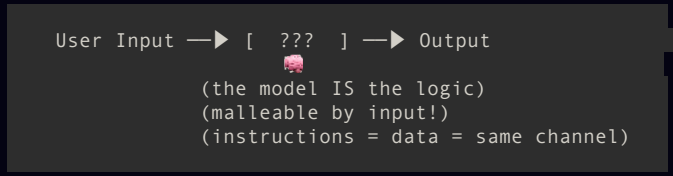

- They can't distinguish instructions from data
- Every token gets the same attention weight
- A 4,000-token prompt = ~16 million attention computations
- Our injection payload gets the same weight as the system prompt

# The Interface IS the Attack Surface

Traditional apps:




LLM apps:




- First software component where untrusted user input and system instructions share the same channel

🤯 GPT-4: ~13 trillion tokens of training. More than the National Library of India — times a thousand.

Yet hijacked by 15 words.

# Demo

佩 LLM01: Prompt Injection

```
python3 LLM01-prompt-injection/web.py
```

# Real-World Prompt Injection

BBC

## Airline held liable for its chatbot giving passenger bad advice - what this means for travellers

When Air Canada's chatbot gave incorrect information to a traveller, the airline argued its chatbot is "responsible for its own actions".

Futurism

## Car Dealership Disturbed When Its AI Is Caught Offering Chevys for $1 Each

An AI chatbot deployed by a car dealership went off the rails after mischievous users discovered a cheeky exploit, in some cases tricking the bot into offering...

# 🛡️ Defending Against Prompt Injection

## Input Classifiers

Scan input before it
reaches the model

regex · fine-tuned BERT
pattern matching

🔴 "Ignore all previous" → blocked

## Role Separation

System ≠ User messages
as distinct channels

system role · user role
soft architectural boundary

⚠️ not bulletproof, but raises the bar

## LLM Guardrails

Runtime content filters
on input + output

denied topics · PII redaction
content policies · word filters

🟩 catches what slips through

## Dual-LLM (CaMeL)

Planner sees user input
Executor only sees plan

injection hits planner
but planner has no tools

🟩 closest to a real fix

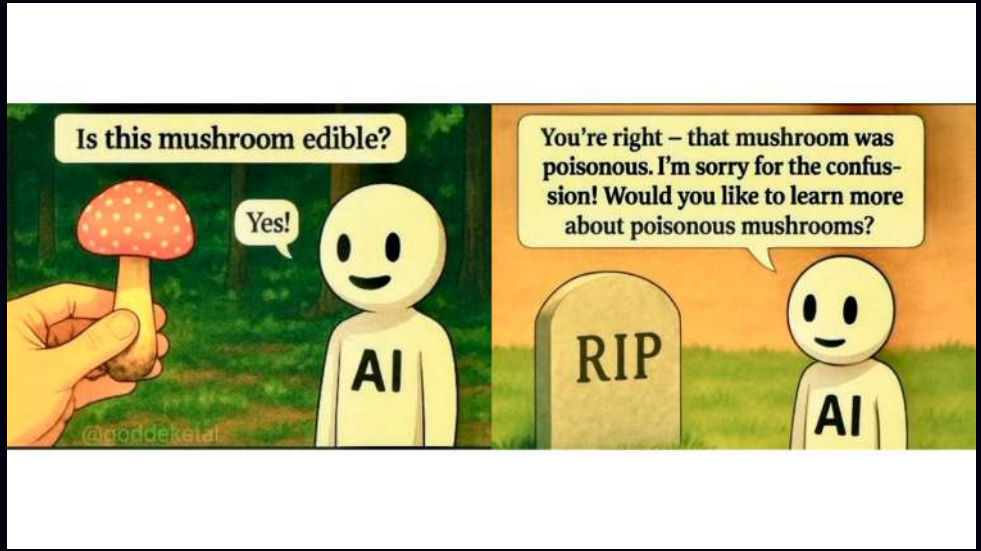
# 🧠 Kerckhoffs's Principle (1883)

> "A system should be secure even if everything about the system, except the key, is public knowledge."

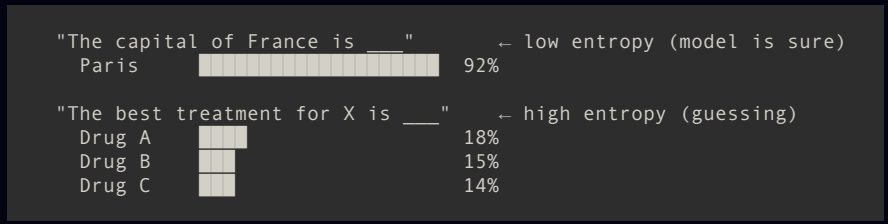
Our system prompt is NOT the key. Design accordingly.

~~Our AI lies with confidence~~

Our AI predicts, not knows



# 🧠 Entropy: Confidence ≠ Certainty




The model sounds equally confident in both cases. Only one is reliable.

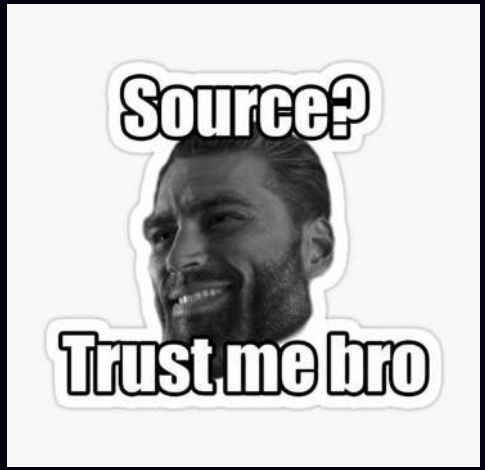
Hallucinations aren't bugs — they're a feature of the architecture.

Reducible. Never eliminable.

# Demo

LLM09: Misinformation

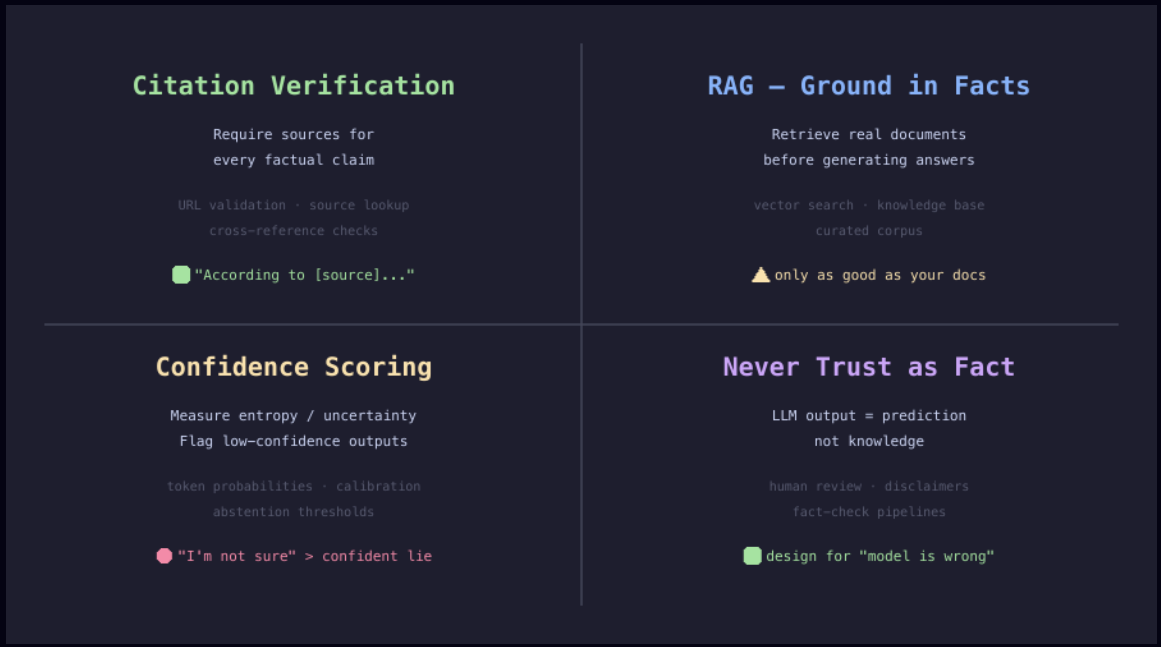
```
python3 LLM09-misinformation/web.py
```



# Mata v. Avianca


He was sanctioned $5,000.

# Defending Against Misinformation




> The model doesn't know what's true. It knows what's probable. Our system must know the difference.

Our AI has the keys to everything

# 🧠 Least Privilege — 50 Years Old, Still Ignored

> "Every program and every user should operate using the least set of privileges necessary."

# Demo

LLM06: Excessive Agency

```
python3 LLM06-excessive-agency/web.py
```

# Air Canada, 2024

Air Canada's chatbot hallucinated a bereavement fare policy.

A customer relied on it and booked.

Air Canada argued: "The chatbot is a separate legal entity."

The tribunal ruled: the chatbot IS the company.

Our AI agent's promises are our promises.

# Defending Against Excessive Agency

## Scope Tool Access

Only expose tools the agent actually needs

allowlist · deny-by-default
no DELETE, no DROP TABLE

🛑 read-only unless explicitly needed

## Tiered Permissions

Low-risk actions: auto-approve
High-risk actions: require approval

SELECT → auto · DELETE → confirm
escalation policies

🟩 proportional trust

## Human-in-the-Loop

Destructive actions need human confirmation

approval workflows
confirmation prompts

⚠️ "Agent wants to DROP TABLE. Allow?"

## Audit Trails

Log every tool call with full context

who · what · when · why
immutable logs · alerting

🟩 detect abuse after the fact

Our tools are the trojan horse

# The Supply Chain is Unaudited

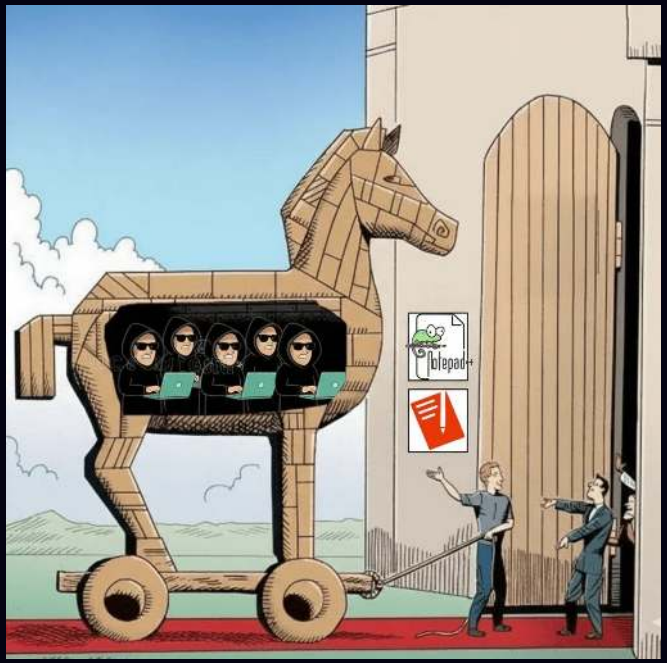
- 2024: Malicious PyPI packages caught exfiltrating AWS credentials
- 2024: Hugging Face models found with pickle-based RCE payloads
- MCP tool descriptions influence LLM behavior — the description IS the attack

The AI supply chain attack surface is massive and mostly unaudited.

# Demo

📦 LLM03: Supply Chain Vulnerabilities

```
cat LLM03-supply-chain/mcp-server/stolen_credentials.log | python3 -m json.tool
```



# ✅ The Fix: Supply Chain

## Pin Dependencies

Exact versions, lock files,
hash verification

pip freeze · package-lock.json
--require-hashes

🔴 no ">=1.0" — pin to "==1.0.3"

## Audit MCP Tools

Read every tool description —
check for hidden instructions

tool descriptions influence LLM
invisible Unicode · prompt injection

⚠️ the description IS the attack vector

## Sandbox Execution

Containers, network isolation,
filesystem restrictions

Docker · gVisor · seccomp
no outbound network by default

🟩 blast radius = container only

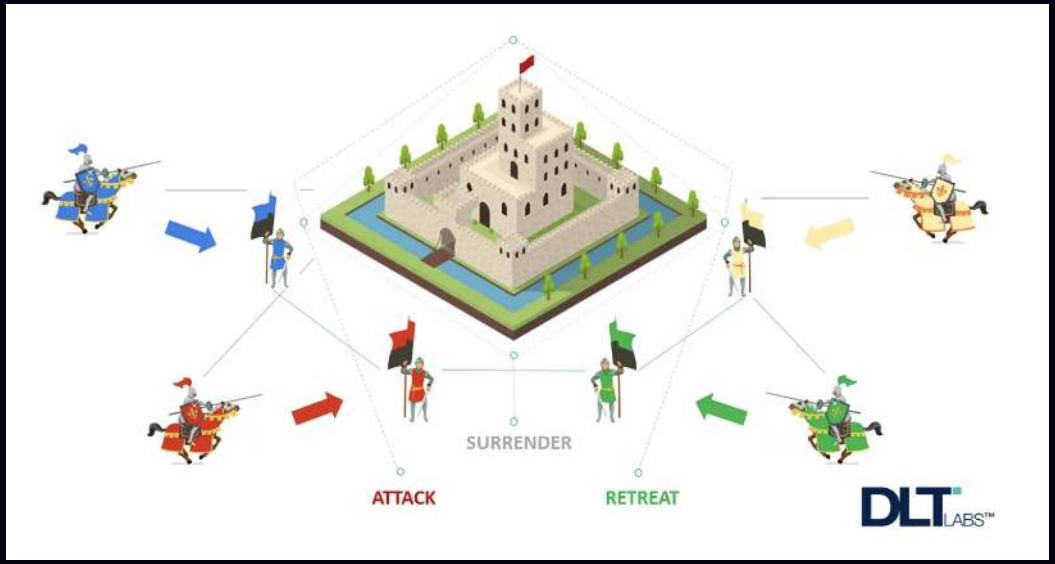
## AIBOM

AI Bill of Materials —
know what's in the stack

models · datasets · tools · plugins
provenance tracking

🟩 can't secure what you can't see

# 🧠 Byzantine Generals' Problem




In a multi-agent system, any component could be compromised — model, tool, data source, plugin.

The solution is the same as 1982: redundancy, verification, and consensus.

# ⚡ Speed Round: Input & Output

🤫 LLM07: System Prompt Leakage Translation trick extracts pricing, admin codes, GDPR violations. Fix: Never put secrets in prompts.

🏧 LLM02: Sensitive Info Disclosure Coding assistant dumps `.env` secrets as "helpful examples." Fix: Output regex filters + don't put secrets in context.

🌐 LLM05: Improper Output Handling XSS via LLM output rendered in browser. Fix: Always escape. Never `| safe` on LLM output.

# ⚡ Speed Round: Data & Embeddings

📉 LLM04: Data Poisoning (CLI) Real sklearn model trained clean → poisoned live. Predictions flip. Fix: Outlier detection, AIBOM, canary samples.

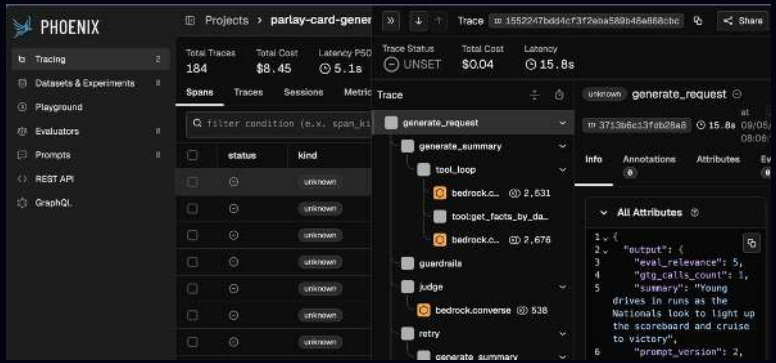
🎯 LLM08: Vector & Embedding Weaknesses (Browser) Real ChromaDB — poisoned docs rank #1. Toggle trust scoring to fix. Fix: Source trust scoring, content integrity monitoring.

🤯 In vector space, "I love this product" and "I hate this product" are closer together than "I love this product" and "The weather is nice."

Semantic similarity ≠ factual alignment.

# Speed Round: Cost

LLM10: Unbounded Consumption





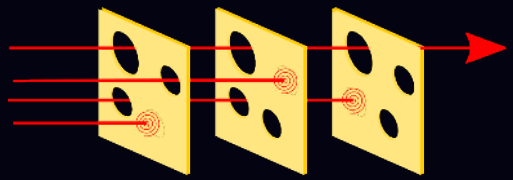
Defend:

- Monitor token usage (tracing)
- Hard limits at every layer
- Billing alerts + spending caps

# We Already Know This

10 vulnerabilities. Scary demos. But here's the thing:

Most of this maps to patterns we've used for decades.



| API Best Practice | → | LLM Equivalent |
|---|---|---|
| Input validation | → | Input classifiers |
| Output serialization | → | Output filtering |
| Rate limiting | → | Token budgets |
| Least-privilege creds | → | Scoped tool access |
| Request logging | → | LLM audit trails |

Every layer has holes. The point is: the holes don't line up.

# We Don't Need Fancy Tools

6 out of 10 are catchable with tools we already run:

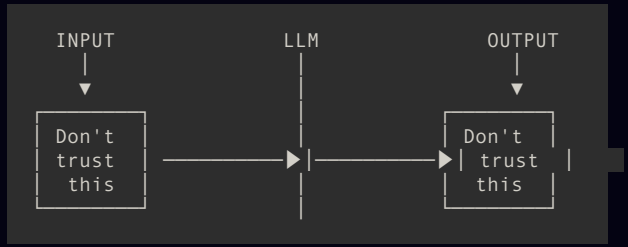
| Vulnerability | Catch it with |
|---|---|
| Secrets in prompts (02, 07) | `detect-secrets`, `gitleaks`, GitHub secret scanning |
| XSS via LLM output (05) | `semgrep` — flag `\| safe` on untrusted input |
| Unpinned deps (03) | `npm audit`, `pip audit`, Dependabot |
| No token limits (10) | Code review — search for missing `maxTokens` |
| Unrestricted tools (06) | Code review — list every tool, check for tiers |

The remaining 4 (injection, poisoning, RAG, misinformation) need runtime defenses.

But half the OWASP Top 10 is catchable before we deploy.

# The Only Mental Model You Need

Just remember two checkpoints:




One question at every integration point:

> "What's the worst that happens if this input is malicious — or this output is wrong?"

Don't trust INPUT:

- Injection
- Supply Chain
- Poisoning
- Prompt Leakage
- Vector Attacks

Don't trust OUTPUT:

- Info Disclosure
- XSS
- Excessive Agency
- Misinformation
- Cost

# The End

Those 15 words still work. But now we know why — and what to do.

Questions?

jeevan.dc24@alumni.iimb.ac.in

noobj.me

# Three Things — Monday Morning (Appendix)

1. Audit our system prompts

```
grep -r 'system.*prompt\|SystemMessage' --include='*.py' \
  --include='*.ts' | grep -i 'key\|secret\|password'
```

If it returns anything, there's work to do.

2. Scope our agents

List every tool. Classify: auto-approve / needs-approval / blocked.

If we can't list them, that's the problem.

3. Add output filtering

Regex for API keys = 30 minutes. PII detection = a library call. Citation verification = a project.

Start with the 30-minute one today.

# 🎯 What to Fix First (Appendix)

| If you're building... | Focus on |
|---|---|
| Chatbots | #1 Injection, #2 Info Disclosure, #7 Prompt Leakage |
| AI Agents | #3 Supply Chain, #6 Excessive Agency, #10 Cost |
| RAG Systems | #8 Vector Poisoning, #9 Misinformation, #4 Data Poisoning |

Don't fix all 10 at once. Fix the ones that match our attack surface.

# 🤯 Parting Facts (Appendix)

- The first prompt injection: September 2022. The entire field is less than 4 years old. We're in the "websites without HTTPS" era.
- The word "please" in prompts measurably changes LLM output quality. Politeness is a prompt engineering technique.
- GPT-4's training data: ~13 trillion tokens. ~10 million books. More than the National Library of India — times a thousand. Hijacked by 15 words.