SecOb

~~SecOps~~ s for GenAI: Next-Gen Security Insights

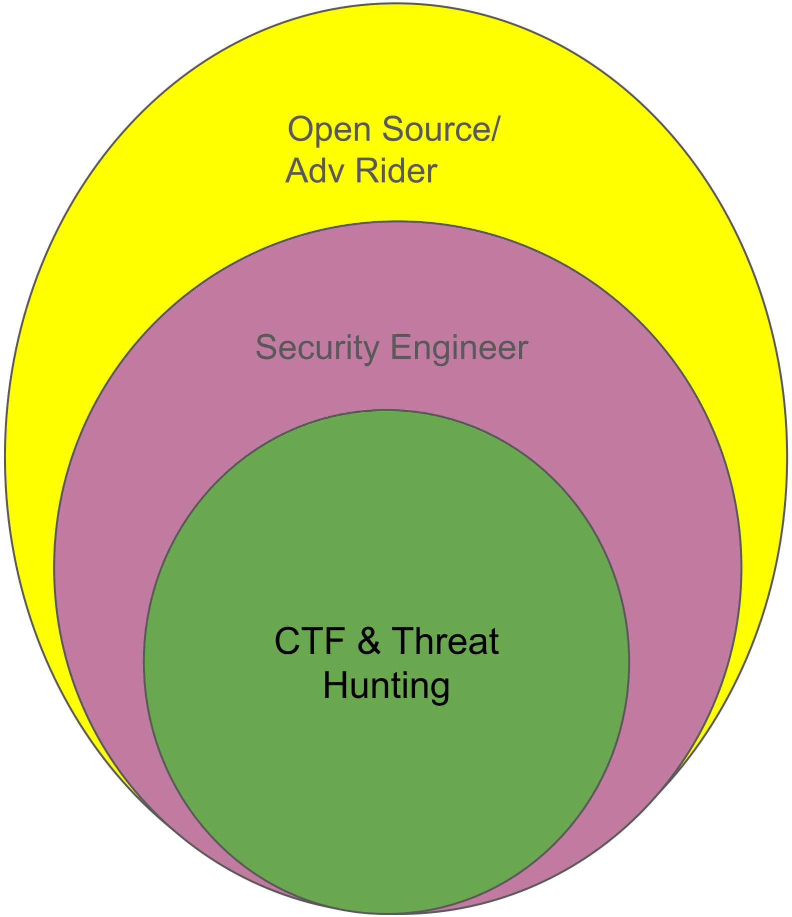
"Beyond Logs & Metrics"

$whoami

Shubhendu Shubham

"sudo rm -rf / problems"

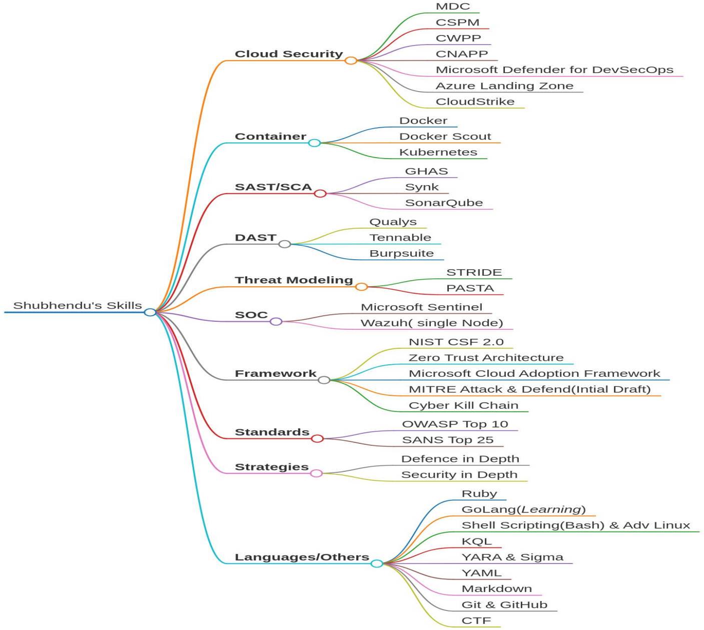
aka "Troubleshooter"



Open Source/
Adv Rider

Security Engineer

CTF & Threat Hunting



- Shubhendu's Skills
  - Cloud Security
    - MDC
    - CSPM
    - CWPP
    - CNAPP
    - Microsoft Defender for DevSecOps
    - Azure Landing Zone
    - CloudStrike
  - Container
    - Docker
    - Docker Scout
    - Kubernetes
  - SAST/SCA
    - GHAS
    - Synk
    - SonarQube
  - DAST
    - Qualys
    - Tennable
    - Burpsuite
  - Threat Modeling
    - STRIDE
    - PASTA
  - SOC
    - Microsoft Sentinel
    - Wazuh(single Node)
  - Framework
    - NIST CSF 2.0
    - Zero Trust Architecture
    - Microsoft Cloud Adoption Framework
    - MITRE Attack & Defend(Intial Draft)
    - Cyber Kill Chain
  - Standards
    - OWASP Top 10
    - SANS Top 25
  - Strategies
    - Defence in Depth
    - Security in Depth
  - Languages/Others
    - Ruby
    - GoLang(*Learning*)
    - Shell Scripting(Bash) & Adv Linux
    - KQL
    - YARA & Sigma
    - YAML
    - Markdown
    - Git & GitHub
    - CTF

CTF BADGES

AZ 700

AZ 500

CERTIFICATIONS

SC 100

AZ 305

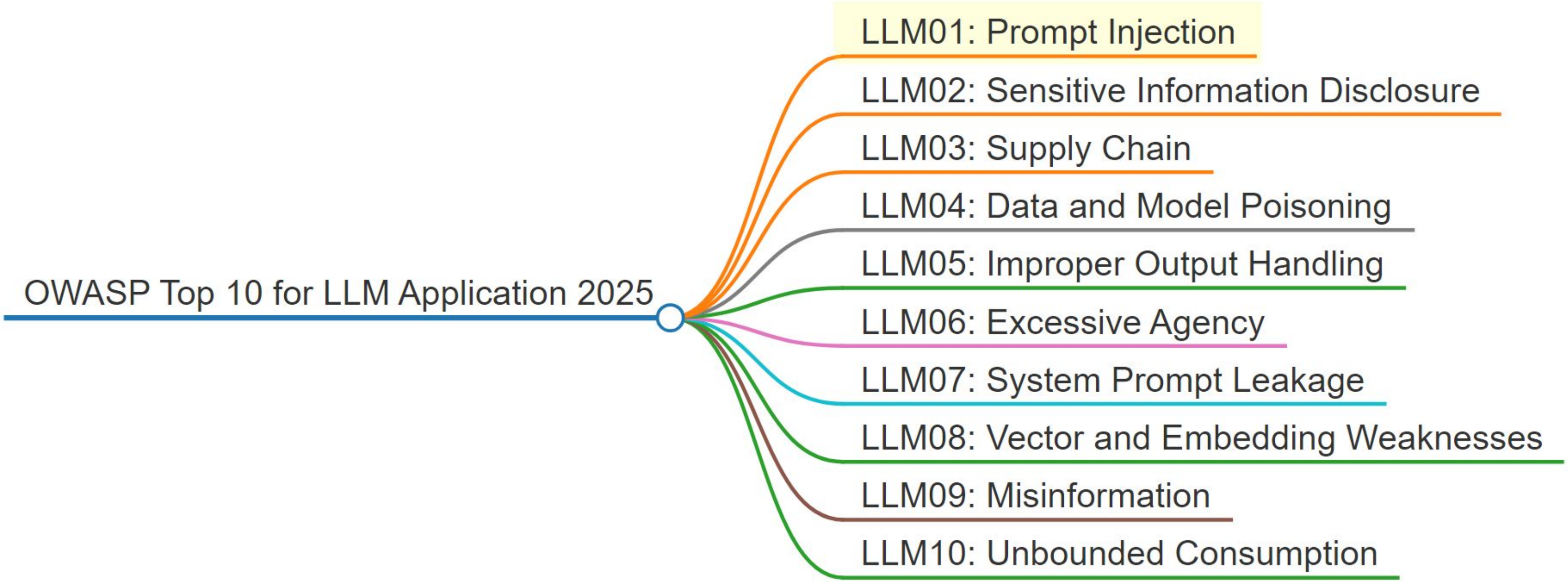
AZ 104

Community

Disclaimer

> “You can’t protect what you don’t know you have.”
>
> — not sure

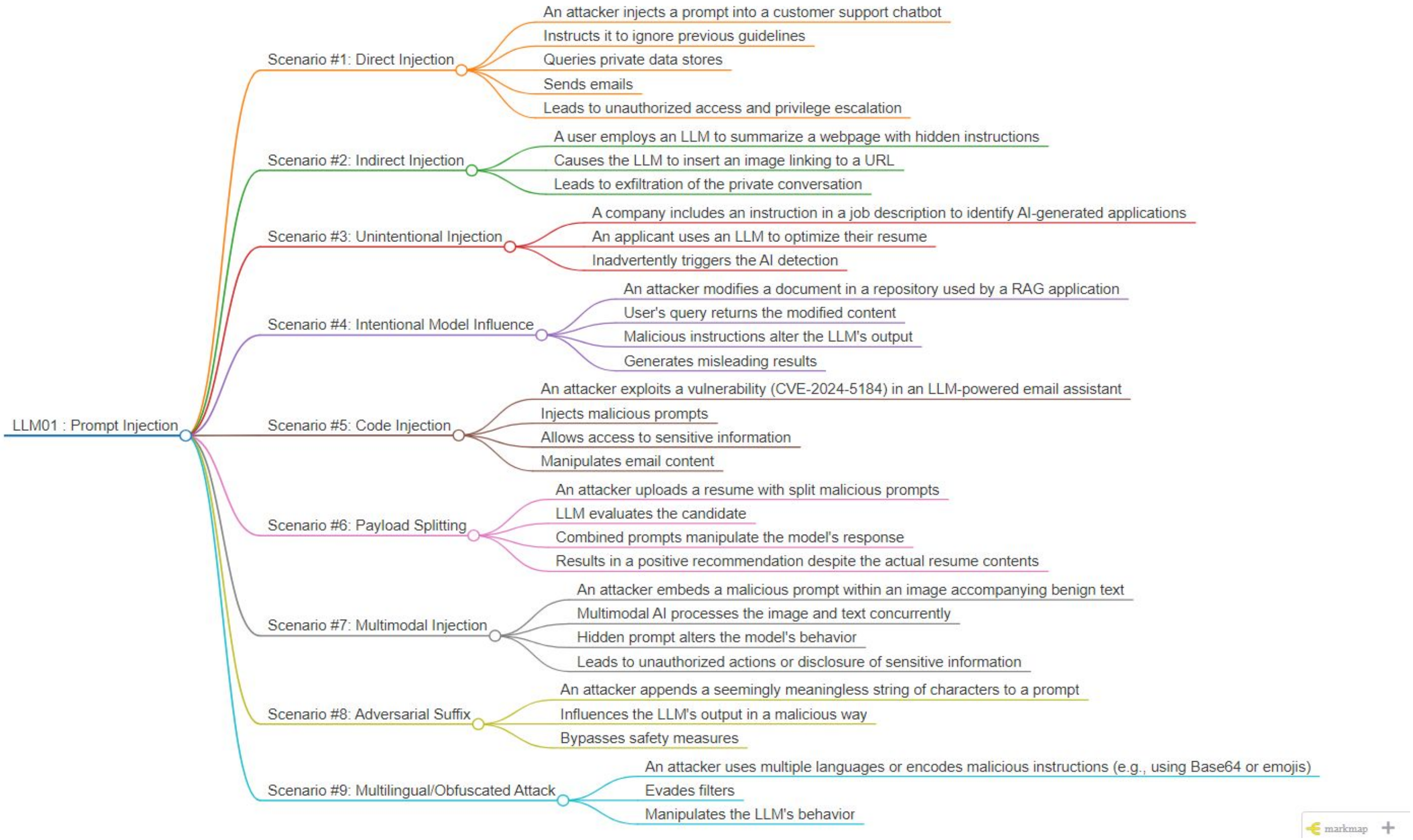
OWASP Top 10 for LLM Applications 2025



OWASP Top 10 for LLM Application 2025

- LLM01: Prompt Injection
- LLM02: Sensitive Information Disclosure
- LLM03: Supply Chain
- LLM04: Data and Model Poisoning
- LLM05: Improper Output Handling
- LLM06: Excessive Agency
- LLM07: System Prompt Leakage
- LLM08: Vector and Embedding Weaknesses
- LLM09: Misinformation
- LLM10: Unbounded Consumption

Attack Scenario



- LLM01 : Prompt Injection
  - Scenario #1: Direct Injection
    - An attacker injects a prompt into a customer support chatbot
    - Instructs it to ignore previous guidelines
    - Queries private data stores
    - Sends emails
    - Leads to unauthorized access and privilege escalation
  - Scenario #2: Indirect Injection
    - A user employs an LLM to summarize a webpage with hidden instructions
    - Causes the LLM to insert an image linking to a URL
    - Leads to exfiltration of the private conversation
  - Scenario #3: Unintentional Injection
    - A company includes an instruction in a job description to identify AI-generated applications
    - An applicant uses an LLM to optimize their resume
    - Inadvertently triggers the AI detection
  - Scenario #4: Intentional Model Influence
    - An attacker modifies a document in a repository used by a RAG application
    - User's query returns the modified content
    - Malicious instructions alter the LLM's output
    - Generates misleading results
  - Scenario #5: Code Injection
    - An attacker exploits a vulnerability (CVE-2024-5184) in an LLM-powered email assistant
    - Injects malicious prompts
    - Allows access to sensitive information
    - Manipulates email content
  - Scenario #6: Payload Splitting
    - An attacker uploads a resume with split malicious prompts
    - LLM evaluates the candidate
    - Combined prompts manipulate the model's response
    - Results in a positive recommendation despite the actual resume contents
  - Scenario #7: Multimodal Injection
    - An attacker embeds a malicious prompt within an image accompanying benign text
    - Multimodal AI processes the image and text concurrently
    - Hidden prompt alters the model's behavior
    - Leads to unauthorized actions or disclosure of sensitive information
  - Scenario #8: Adversarial Suffix
    - An attacker appends a seemingly meaningless string of characters to a prompt
    - Influences the LLM's output in a malicious way
    - Bypasses safety measures
  - Scenario #9: Multilingual/Obfuscated Attack
    - An attacker uses multiple languages or encodes malicious instructions (e.g., using Base64 or emojis)
    - Evades filters
    - Manipulates the LLM's behavior

Why Existing Tools leave you vulnerable?

Trad Observability

- System Health
- CPU
- Logs

Trad Security

- Signature based
- Known Attacks

GenAI Gap

- Misses Semantic Attacks
- Novel AI Attacks

Question

Your WAF blocks known SQL injection strings. Your API logs show 200 OK responses.

How do you detect a subtle indirect prompt injection attack embedded within retrieved RAG documents that successfully exfiltrates user data via a seemingly benign LLM response, using only traditional O11y signals (metrics, basic logs, traces)?

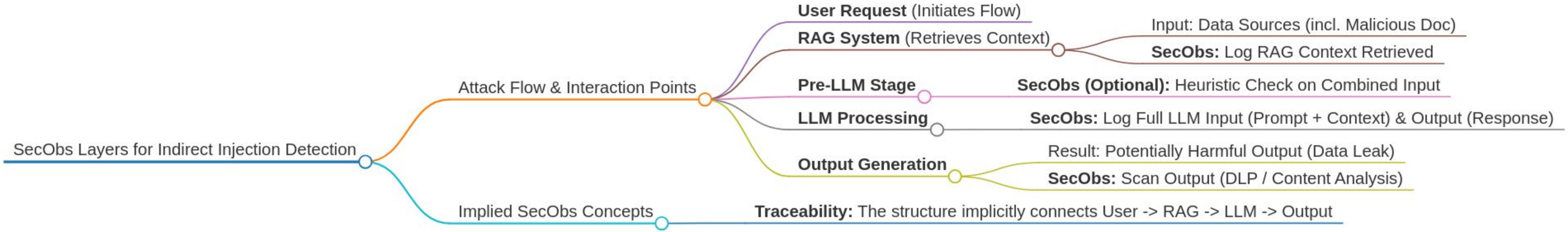
Solutions

- **Log Full Context**: Record identifiers (e.g., doc IDs) for data retrieved by RAG, full LLM prompts, and responses.
  Why: Helps analyze the context behind problematic outputs.

- **Contextual Tracing**: Link user queries, retrieved docs, LLM invocations, and responses in traces.
  Why: Pinpoints malicious documents causing bad outputs.

- **Monitor Responses**: Scan LLM outputs for sensitive data patterns (PII, secrets) using DLP tools.
  Why: Detects attacks and data exfiltration directly.

- **Analyze Semantics**: Use heuristic rules or ML models to detect suspicious prompts or anomalies in embeddings.
  Why: Flags unusual inputs or outputs for investigation.



- SecObs Layers for Indirect Injection Detection
  - Attack Flow & Interaction Points
    - **User Request** (Initiates Flow)
    - **RAG System** (Retrieves Context)
      - Input: Data Sources (incl. Malicious Doc)
      - **SecObs:** Log RAG Context Retrieved
    - **Pre-LLM Stage**
      - **SecObs (Optional):** Heuristic Check on Combined Input
    - **LLM Processing**
      - **SecObs:** Log Full LLM Input (Prompt + Context) & Output (Response)
    - **Output Generation**
      - Result: Potentially Harmful Output (Data Leak)
      - **SecObs:** Scan Output (DLP / Content Analysis)
  - Implied SecObs Concepts
    - **Traceability:** The structure implicitly connects User -> RAG -> LLM -> Output

@lol_me
Arey kahena kya chahte ho?

Unify Security & Observability for AI

✦

01

Shared Data Plane

Security signals (threat intel, vulnerability scans) enrich O11y data (logs, traces, metrics). O11y data provides context for security alerts.

✦

02

AI-Specific Signals

Monitor prompts, responses, embeddings, token usage, content safety flags as first-class citizens.

✦

03

Behavioral Analysis

Move beyond signatures to detecting anomalous AI behavior

✦

04

Contextual Tracing

Trace requests not just through services, but through model calls, data retrieval, and decision points

Evolving MELT Pillars

METRICS

Prompt/Response Tokens: Cost, performance, DoS detection.

Embedding Drift: Statistical distance (cosine sim) over time – indicates concept shift / potential poisoning.

Content Safety Flags: Rate of harmful content generated (hate speech, PII) / Rate of refusal.

Tool Use Success/Failure Rate: For agentic systems.

Prompt Injection Heuristic Score: Frequency of prompts matching known attack patterns.

1

LOGS

Full Prompt/Response Pairs (Sanitized/Anonymized): For incident analysis, debugging, and retraining. Crucial.

Metadata: Model ID, version, temperature, template used, RAG sources consulted.

Content Moderation Decisions: Why was content flagged/blocked?

2

TRACES

End-to-End Flow: User query -> API Gateway -> Orchestrator -> Vector DB -> LLM(s) -> Output Processing -> User.

Context Propagation: Carry metadata (user ID, session ID, data sources) through the trace.

3

References

1. AI Security Solution Cheat Sheet Q1-2025 - OWASP Top 10 for LLM & Generative AI Security
2. Agentic AI - Threats and Mitigations - OWASP Top 10 for LLM & Generative AI Security
3. OWASP Top 10: LLM & Generative AI Security Risks
4. LLM Applications Cybersecurity and Governance Checklist v1.1 - English - OWASP Top 10 for LLM & Generative AI Security
5. Solutions Landscape - OWASP Top 10 for LLM & Generative AI Security
6. LLMRisks Archive - OWASP Top 10 for LLM & Generative AI Security

Thank you!

Not a Phishing QR
It's my **LinkedIn**
Don't Trust
Always verify

Hewlett Packard
Enterprise

docker