Building Lightweight Security Tools Using GenAI

Presented by: Ritika Verma & Ashwin Iyer

Why GenAI for Security?

Strength

📄 Summarizing Unstructured Data

🧠 Natural Language Understanding

⚙️ Automation Superpowers

When It Fails

❗ Hallucinates

❌ Not Deterministic

🧱 Requires Guardrails

GenAI and Security: Exploring Opportunities

Blue Team

- Alert Enrichment, IOC Extraction, Log Analysis

Red Team

- Payload Obfuscation, Phishing Content Generation

DevSecOps

- Code scanning, Secure config suggestions

Threat Intel

- Report Summarization, Actor profiling

Automation Superpowers

- Speed, Scale and Context aware

Ethical and Practical Considerations

Risks of Using GenAI in Security Workflows

- **Prompt Injection (LLM01)**
- **Insecure Output Handling (LLM02)**
- Training Data Poisoning (LLM03)
- **Model Hallucination (LLM06)**
- **Sensitive Information Disclosure (LLM04)**
- **Overreliance Without Oversight (LLM07)**

Secure-by-Design Patterns

- Input Filtering
- Scoped Prompts
- Output Validation
- Human-in-the-loop

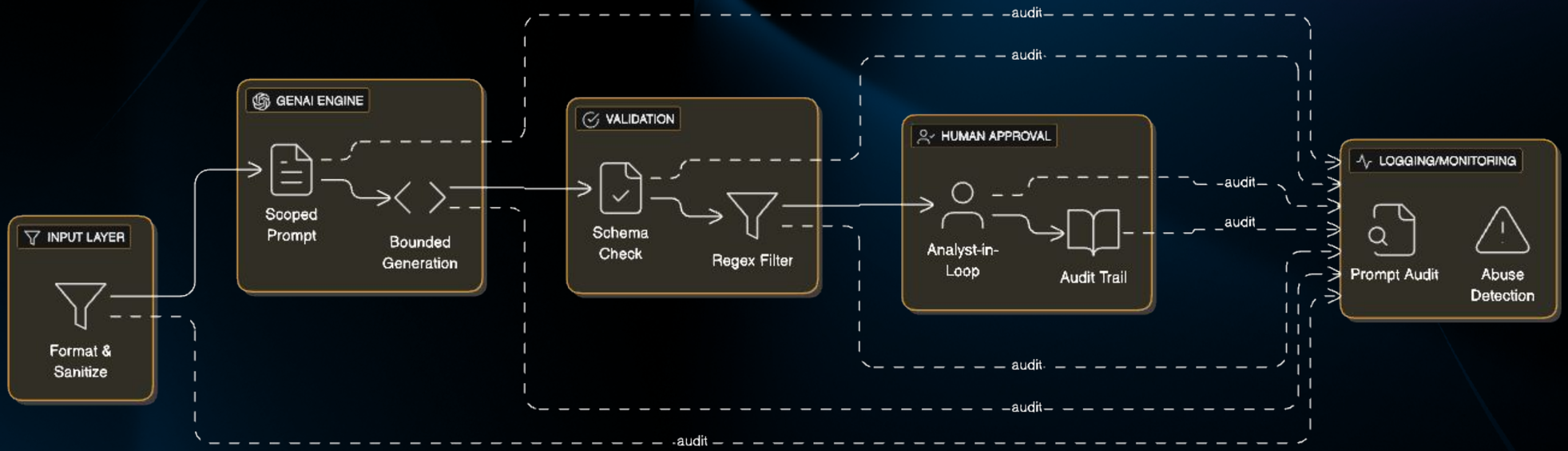
Logging & Monitoring

- Track prompt-response history
- Detect abnormal usage
- Indicators of jailbreaking.

Secure GenAI System Design

1. **Input Pre-processing:** Sanitize logs, extract targets
2. **Scoped Prompting:** No open-ended generation
3. **Validation:** Schema + Regex filters
4. **Output Handling:** Logging, HITL escalation
5. **Monitoring:** Abuse detection & prompt audit trail

Secure GenAI Architecture



eraser

Prompting: Do It Right or Risk It All

✅ **Good Prompts:**

🧠 Be specific about your task and expected format

🗒️ Provide context: logs, code, reports, goals

🛠️ Use step-by-step instructions

📋 Ask for structured output (JSON, table, bullet list)

❌ **Bad Prompts:**

❓ Vague commands like “analyze this”

🌀 No context or too much irrelevant info

🎲 Open-ended with no constraints

🚨 Asking for security analysis without defining scope

Lab 1: Hands-On

Lab 1 – Prompt Engineering

Structured prompts

Lead to predictable, machine-parseable results.

Downstream automation

Useful for ticket generation, dashboard updates, or rule generation

Safety and Reliability

Promotes LLM safety and reliability by controlling input/output.

Lab 1 – Try It Yourself

- 🧪 Compare 2 prompts using the same log snippet or code
- 🧠 Which one gives clearer, more useful results?
- 🛠️ Modify the bad prompt to improve it
- 🔁 Try a structured format (JSON/table) vs free text

Lab 2: Hands-On

Lab 2 – Phishing Email Analyzer

📥 Input: Raw email content (subject, headers, body text)

🧠 GenAI Task:

Classify email (Phishing/Benign), extract indicators (URLs, sender domain), check with VirusTotal

📤 Output:

Structured threat profile (JSON or table), risk score, recommended action (alert / allow / escalate)

🎯 Goal:

Automate first-pass triage for SOC analysts → reduce false positives, prioritize review queue

💡 Built with:

LangGraph (agentic logic) + OpenAI + VirusTotal API + Python filter function

Lab 2 – Prototyping Security Helper Tool

Building Multi-Agent Security System

- Email Analyzer Agent
- Threat Intelligence Agent
- URL Reputation Agent
- Decision Coordinator Agent

Technologies

- LangGraph for agent orchestration
- OpenAI GPT-4 for intelligent analysis
- VirusTotal API for threat intelligence

Shared State Management

- EmailSecurityState flows through all agents
- Progressive Analysis
- Collective Intelligence

Lab 2 – Try It Yourself

Things to Try

🔁 Workflow Variants:

- Parallel agents (URL check + LLM)
- Add HITL or guardrails step

✍️ Prompt Variations:

- “Why might this be phishing?”
- Few-shot examples of known phishing to increase confidence

🔗 Other External Checks:

- WHOIS domain age lookup

From Prompt to Protection: Full Lifecycle Guardrails

🛡️ 1. Input Protection

Prompt validation (regex / LMQL)

PII masking & tokenization

Role- & scope-based input control (MCP)

🧠 2. LLM Execution

Guardrails: schema & rule enforcement

Output validation (LangChain / DSPy)

Hallucination & unsafe output rejection

🔍 3. Output Filtering

Banned pattern checks (regex)

Confidence threshold gating

HITL escalation for edge cases

📈 4. Monitoring & Drift

Prompt logs (PromptLayer / LangSmith)

Response drift detection

Abuse/jailbreak alerting

🔒 5. Secrets & Runtime Security

Secure key storage (.env / Vault / Doppler)

No hardcoded creds

Scoped permissions via agent policies

OpenSource Tools

1. 🛡️ Input Protection
Prompt Validation: Guardrails AI, LLM Guard, LangSmith
PII Masking: Microsoft Presidio, spaCy NER, Faker (for anonymization)

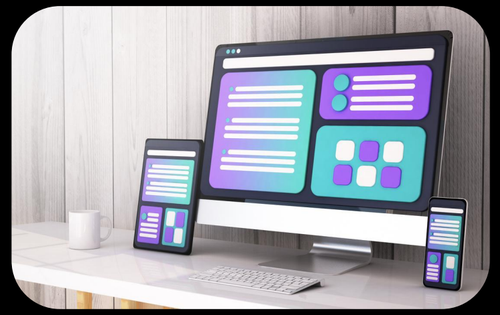
2. 🧠 LLM Execution
Schema Enforcement: Guardrails AI , Pydantic , Jsonschema
Output Validation: NVIDIA NeMo Guardrails
Hallucination Detection: SelfCheckGPT , FActScore , Custom fact-checking: LLM / RAG Evaluation

3. 🔍 Output Filtering
Content Filtering: Detoxify
Review Workflows: Label Studio

4. 📊 Monitoring & Drift
Logging: MLflow, TensorBoard

5. 🔐 Secrets & Runtime Security
Key Management: HashiCorp Vault (OSS)
Runtime Security: OWASP ZAP, Bandit (code scanning)

Workshop Reflection



Prompt Engineering

✔ Extract CVEs, IOCs from noisy logs
✔ Secure prompt formatting (JSON schemas)
✔ Compared vague vs structured prompts

Security Tool Prototype

✔ Built phishing classifier with LangGraph agents
✔ Added external checks (VirusTotal, heuristics)
✔ Used LangChain tools for explainability

Skills Gained

✔ Security-aware prompt design LangGraph multi-agent orchestration
✔ Guardrails integration & safety enforcement
✔ Threat intelligence API chaining
✔ Evaluating LLM outputs for hallucination/misuse

Q&A and Feedback

- Q&A
- Group photo

Thank You!