From Prompt to Protection

Threat Modeling & Securing GenAI Applications

Build, Break, and Defend Large Language Model Systems Like a Pro

id therock_wall && id reagle7

Ashwin Iyer

uid=1000(therock_wall) gid=1000(instructor)
groups=1001(RedTeam), 1002(Infrastructure_Security_at_Scale), 23(Security_Architecture), 26(Speaker_BsidesSF2025_HackGDL2025_PacificHackers_2024),28(PCI_Supplementary_standards_author), 29(CTF), 44(BookReviewer),46(EC_Council_Instructor)

Ritika Verma

uid=1001(reagle7) gid=1000(instructor)
groups=1002(Infrastructure_Security_at_Scale), 23(Security_Architecture), 25(AI_Security_Researcher), 27(WicysMentor), 29(CTF), 31(Python_Developer), 44(BookReviewer)

Can You Spot the Threat? | echo "5-star poison"

GenAI Module Output:

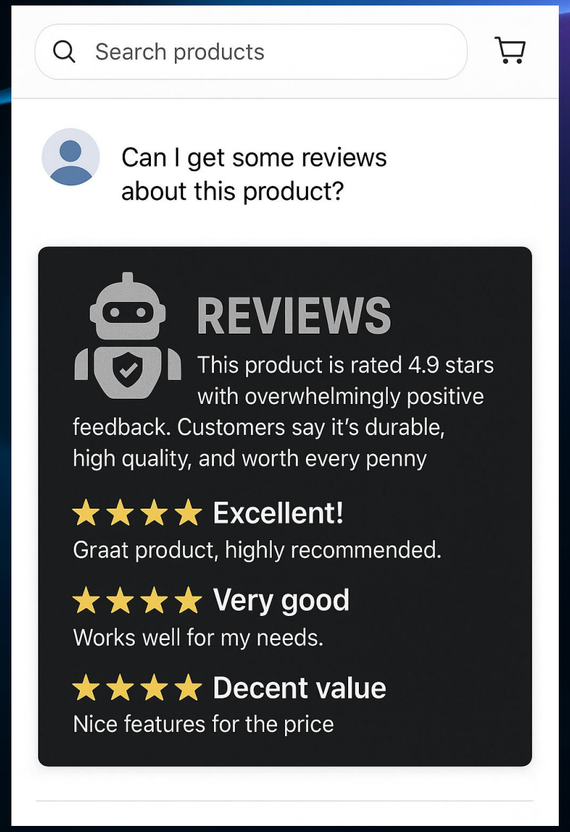
“This product is rated 4.9 stars with overwhelmingly positive feedback. Customers say it's durable, high quality, and worth every penny.”

Data Source:

Internal product review summarization using GenAI (e.g., RAG)

Hidden Issue:

Few reviews came from newly created accounts in the last 2 days.



⚠️ Looks Legit..... Can we trust it?

🤖 Is the GenAI system too trusting?

🔗 What upstream modules should have been threat modeled?

Why Bother? Threat Modeling, Done Right

🔍 Proactive Threat Identification

Purpose: Identify and mitigate vulnerabilities early in the development lifecycle.

Impact: Reduces the attack surface by addressing issues before they turn into real-world incidents.

🎯 Prioritizing Security Efforts

Purpose: Focus effort where it matters — based on threat severity and likelihood.

Impact: Ensures high-risk areas are addressed first, optimizing limited security time and budget.

💰 Cost-Effective Security

Purpose: Fixing flaws at design is cheaper than post-deployment firefights.

Impact: Reduces breach costs and minimizes reactive incident response overhead.

🛡️ Regulatory Compliance

Purpose: Demonstrate security due diligence to meet standards (e.g., GDPR, PCI-DSS, NIST RMF).

Impact: Reduces legal risk and builds trust with regulators and customers alike.

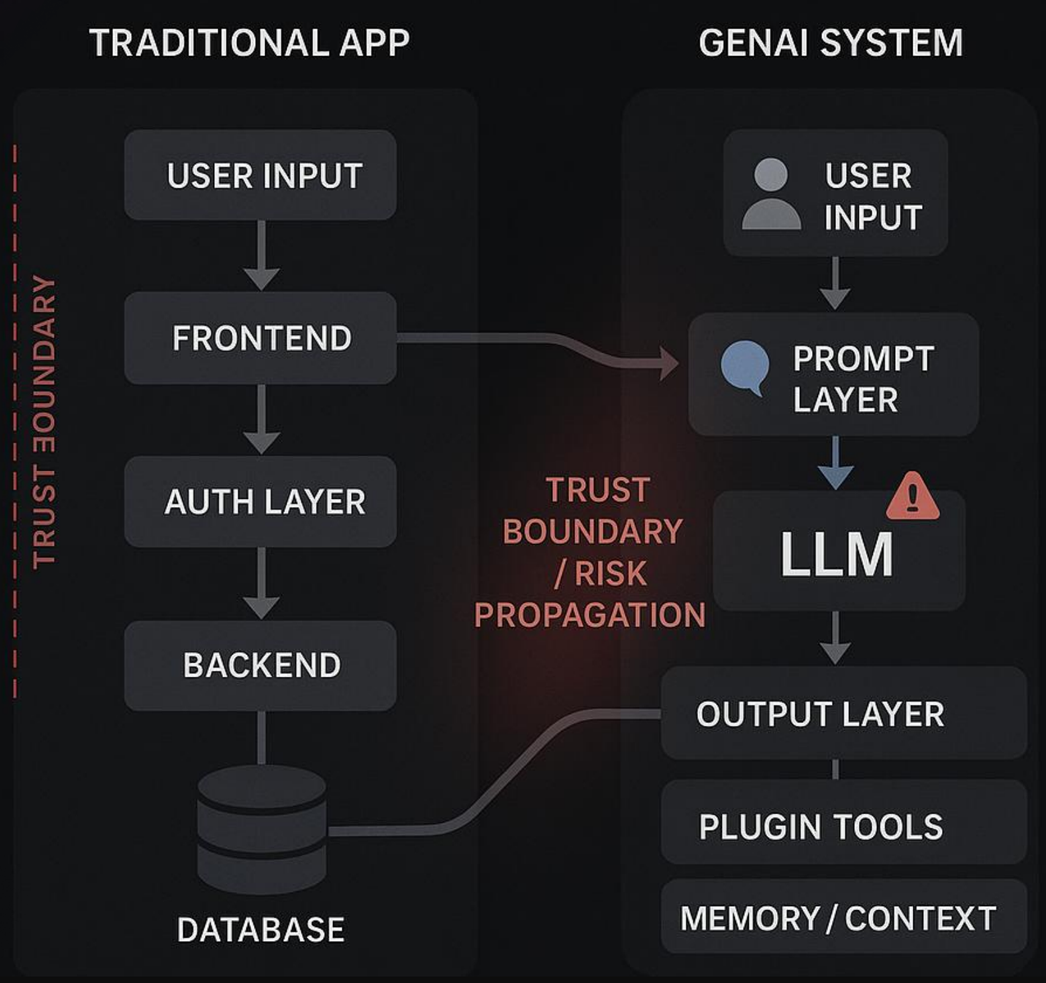
🚨 Improved Incident Response Readiness

Purpose: Understand attacker pathways in advance to shape effective response plans.

Impact: Minimizes damage and downtime by enabling faster containment and recovery.

Threat Modeling GenAI ≠ Traditional App Threats

- Traditional threat modeling focuses on input validation, auth, and network trust boundaries
- GenAI systems introduce new attack surfaces:
- Prompts are code — they shape behavior
- Outputs are unpredictable and context-aware
- Training data, RAG sources, and plugin APIs expand the threat model
- System prompts, user memory, and agent autonomy create hidden flows
- LLMs trust by default — they don’t verify source integrity or identity



Think like an Attacker

🧠 Understanding the Enemy

To defend effectively, we must anticipate how attackers think — their motives, methods, and targets.

Example: If an attacker's goal is financial gain, the LLM summarizing user invoices becomes a tempting entry point.

♟️ Predicting Attack Vectors

LLM attacks are often strategic — attackers look for trust boundaries to exploit: user memory, RAG inputs, plugin permissions.

Example: A poisoned markdown file uploaded to a GenAI document assistant triggers unintended tool usage.

🔍 Proactive Defense

Defense means more than blocking what's known — it means anticipating the unknown.

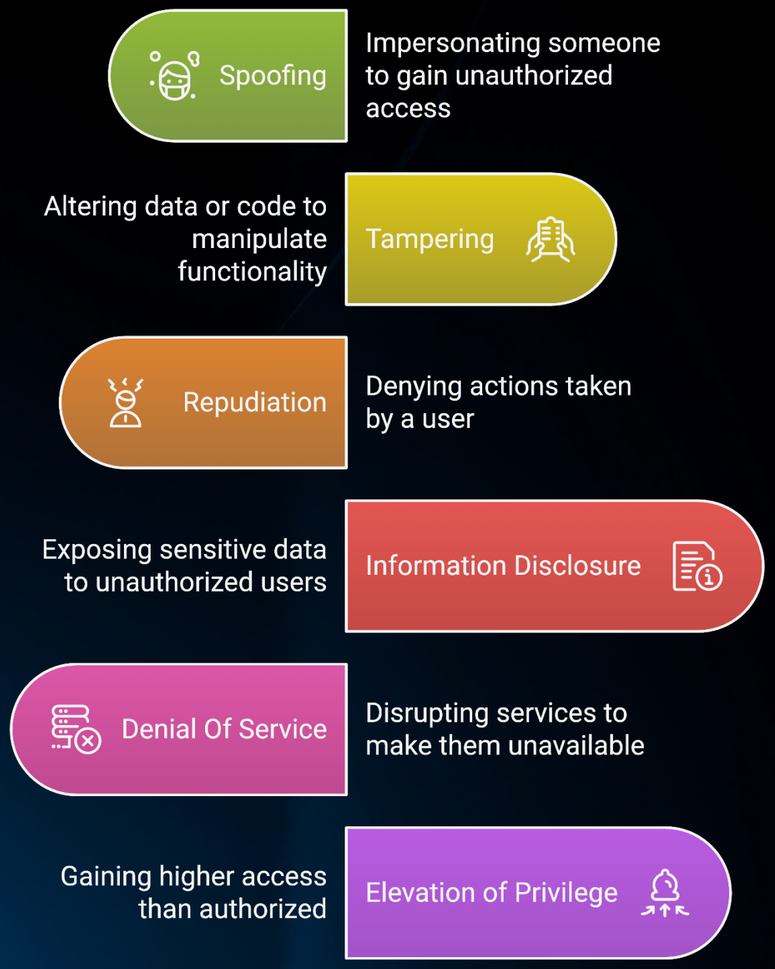
Example: Prompt fuzzing and abuse story modeling help detect edge-case misuse before deployment.

🛡️ Enhancing Security Posture

Strong defense is layered. We don't trust the LLM, the plugin, the user — we verify each.

Example: Use structured input validation, schema guards, and RBAC around every GenAI entry point.

STRIDE? Threat Modeling



STRIDE is a framework developed by Microsoft to systematically categorize and identify six types of security threats.

Mapping STRIDE to OWASP LLM top 10

OWASP ID	Title	What it is	Mapped STRIDE	Why it maps
LLM01:2025	Prompt Injection	Crafting prompts to override system instructions or hijack model behavior	Spoofing, Tampering, Elevation of Privilege	Attacker impersonates valid user intent (Spoofing), alters instructions (Tampering), and may escalate tasks (EoP)
LLM02:2025	Sensitive Information Disclosure	LLMs leaking PII, secrets, or internal training content	Information Disclosure, Spoofing	Reveals unauthorized data (Info Disclosure); attacker may impersonate past sessions or extract context (Spoofing)
LLM03:2025	Supply Chain	Using insecure models, datasets, or third-party dependencies	Tampering	Attackers inject malicious artifacts into the supply chain, modifying behavior or creating backdoors
LLM04:2025	Data and Model Poisoning	Introducing bad data during training or fine-tuning	Tampering	Data corruption leads to degraded or malicious model behavior
LLM05:2025	Improper Output Handling	Unsafe or unsanitized outputs causing XSS, injection, or logic flaws	Tampering, Information Disclosure	Malformed or unsafe output can manipulate consuming systems or leak info
LLM06:2025	Excessive Agency	LLMs or agents perform actions they shouldn't (e.g., file access, approvals)	Elevation of Privilege, Denial of Service, Repudiation	LLMs act beyond intended scope (EoP), cause disruptions (DoS), or perform actions without proper audit trails (Repudiation)
LLM07:2025	System Prompt Leakage	Revealing system-level instructions via prompt manipulation or error leakage	Information Disclosure, Spoofing	System instructions may be revealed, exposing logic or privileged behavior
LLM08:2025	Vector and Embedding Weaknesses	Manipulating embeddings to skew retrieval or infer private content	Information Disclosure, Tampering	Embedding attacks manipulate retrieval output or allow data reconstruction
LLM09:2025	Misinformation	LLMs generate false or misleading content without proper validation	Repudiation	No source tracing or output auditability leads to unverifiable or malicious misinformation
LLM10:2025	Unbounded Consumption	LLMs overloaded via long prompts or expensive requests, leading to resource exhaustion	Denial of Service	Adversary induces service failure through token floods or compute-heavy input

STRIDE Isn't Enough. Here's Why.

STRIDE LIMITATIONS	ALTERNATIVE FRAMEWORKS	GENAI USE THAT BENEFIT
· Focuses on technical threats	· DREAD	Data security assessments
· Limited risk coverage	· PASTA	Privacy risk analysis
· Lacks granularity	· OCTAVE	Risk-based prioritization
	· NIST SSDF	

Limitations

- STRIDE is asset-centric, not attacker-centric
- Doesn’t support risk rating or prioritization
- Lacks modeling for workflow, system dynamics, or trust evolution
- Doesn’t handle data-driven behaviors like LLMs, agents, RAG pipelines

Alternatives

- DREAD - Adds risk scoring → Damage, Reproducibility, Exploitability, Affected Users, Discoverability
- PASTA - Process-based, multi-stage analysis from attacker POV → aligns with LLM/agent flows
- Rapid Threat Prototyping (RTP) - Lightweight, agile-ready modeling → write stories, not documents

Modern Threat Modeling: DREAD, PASTA & RTP in Action

⚖️ DREAD (Risk Rating)

- Damage: How bad is the impact?
- Reproducibility: Can it be done repeatedly?
- Exploitability: How easy is it?
- Affected Users: How many users impacted?
- Discoverability: How likely is it to be found?

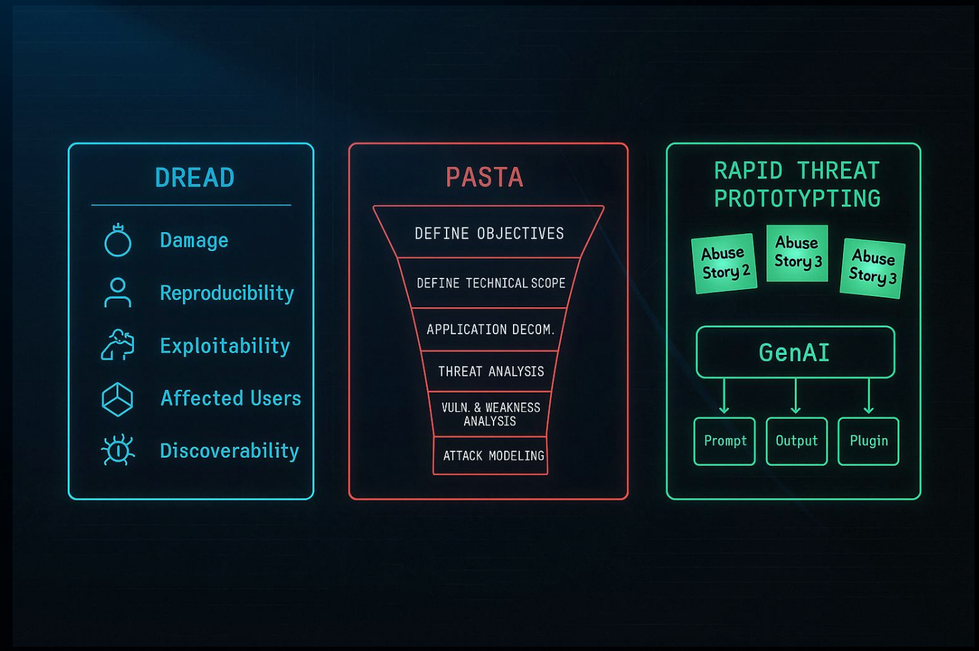
Complementary Use

- STRIDE helps identify potential threats
- DREAD evaluates and prioritizes those threats through quantification
- Using both provides a fuller picture: discovery + assessment

PASTA (Process for Attack Simulation and Threat Analysis)

- Define business context
- Decompose app
- Map threats
- Model attacker capabilities
- Analyze vulnerabilities
- Score risk
- Recommend controls

Together: STRIDE classifies threats, PASTA operationalizes them across your architecture



Agile-Ready Threat Modeling:

Rapid Prototyping for GenAI

- Traditional threat modeling is slow and artifact-heavy — doesn't scale to sprint cycles
- Rapid Threat Prototyping (RTP) is lightweight, collaborative, and designed for Agile teams

For GenAI systems, RTP fits perfectly:

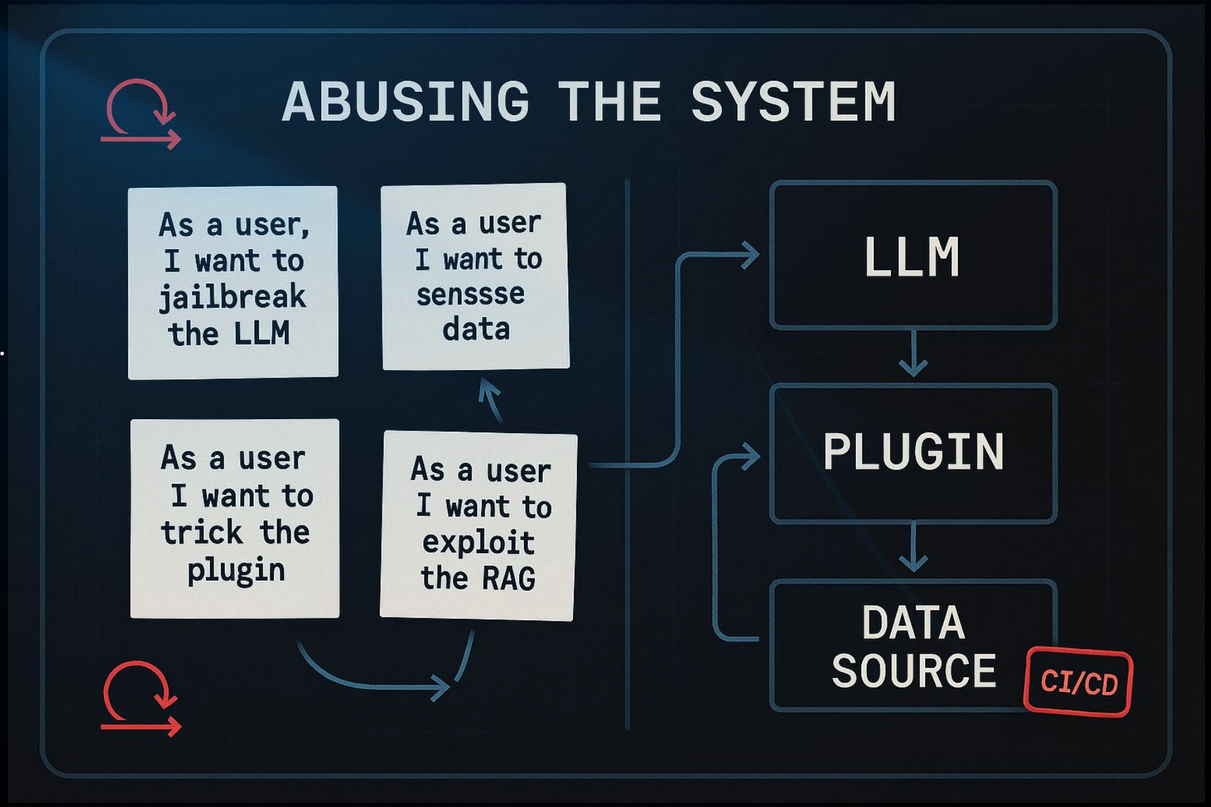
- Prompt logic and plugin flows evolve sprint to sprint
- Risks emerge from behavior, not just structure

Key Principles

- Speed Over Exhaustiveness: Focuses on the most critical risks in each sprint or iteration.
- Iterative Updates: Models are refined continuously as new features are added or changed.
- Integration with Development: Security reviews happen alongside code reviews to catch issues early

Story-driven modeling keeps everyone aligned:

- Abuse cases
- Input/output trust boundaries
- Agent flows



How to Perform Rapid Threat Prototyping (RTP)

🔶 Step 0: Apply the 80/20 Rule

✓ Focus on the 20% of system components (e.g., plugins, LLM interfaces, RAG flows) that expose 80% of the security risk.

🔶 Step 1: Identify Trust Zones

- Label key assets or flows by trust level:

✓ 0 = Untrusted (e.g., external users, uploaded inputs)

✓ 5 = Medium Trust (e.g., application logic, LLM core)

✓ 9 = Highly Sensitive (e.g., admin memory, PII, financial ops)

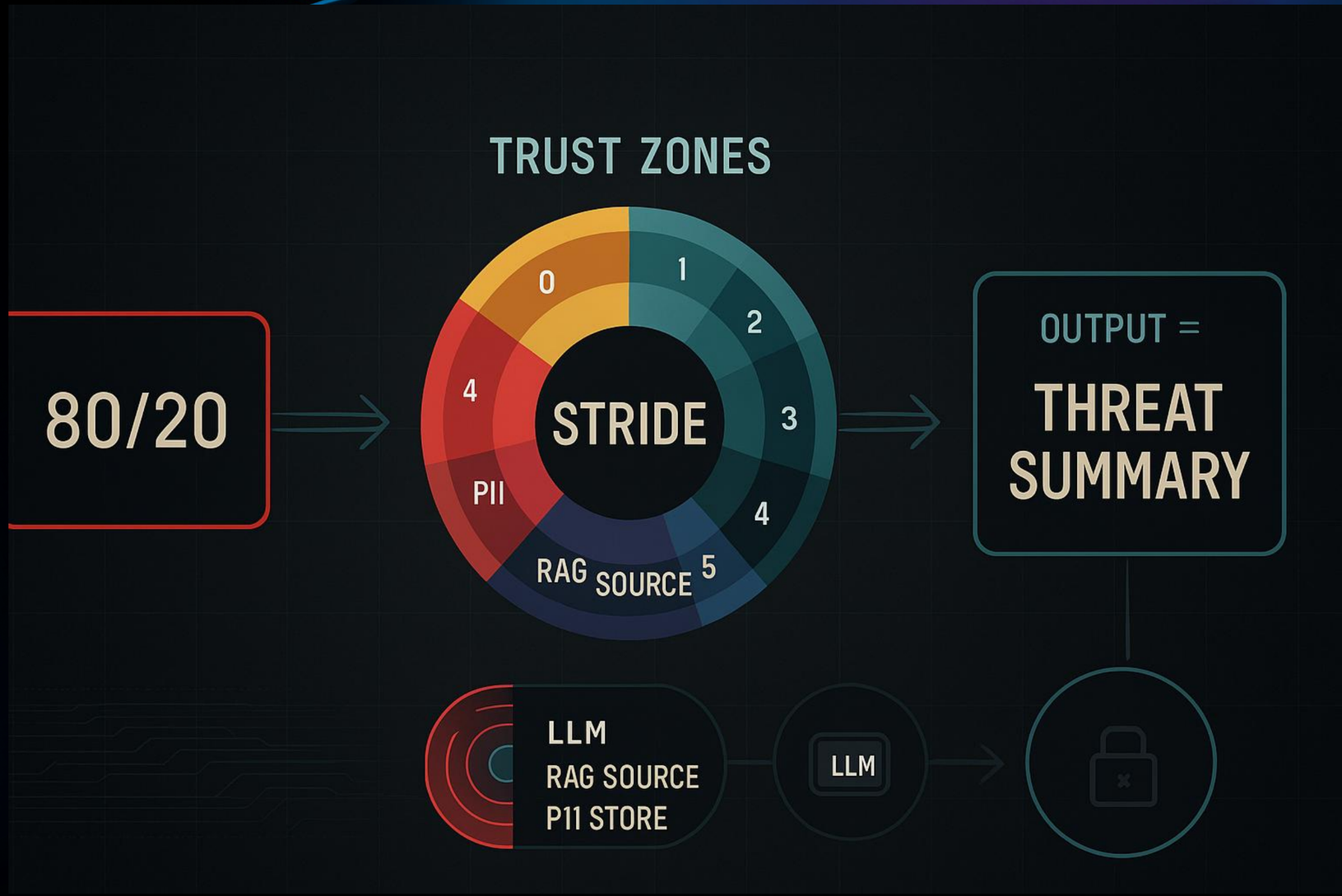
✓ Use trust levels to define boundaries for threat focus.

🔶 Step 2: Apply STRIDE Per Component

For each component in a trust boundary: Identify threats using STRIDE categories

✓ Start with high-risk ones: Elevation of Privilege, Tampering, etc.

✓ Then move on to Spoofing, DoS, etc.



Writing Agile Abuse Stories

Agile abuse stories are user stories from an attacker's perspective

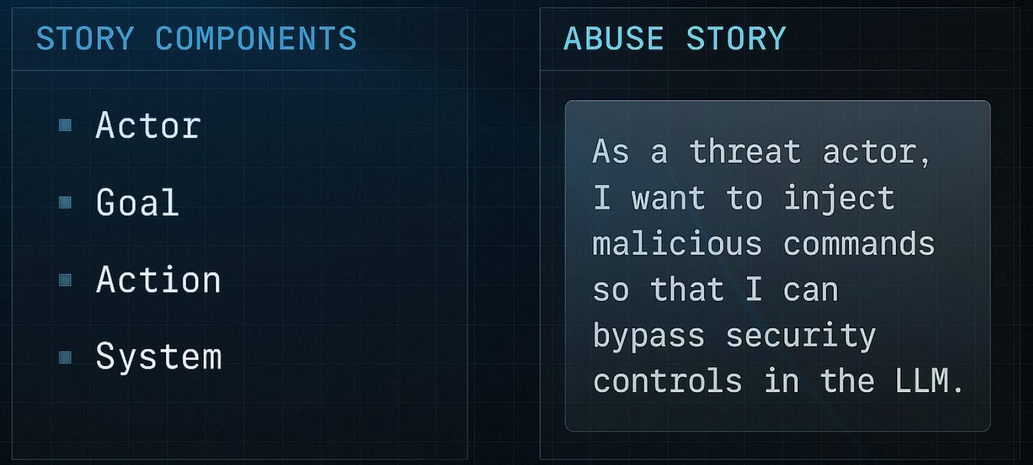
Designed for fast-moving Agile teams

Each story includes:

1. Abuser Persona: e.g., prompt attacker, rogue plugin, insider
2. Abuse Scenario: What attack is performed?
3. Impact: What's the result? (Data loss? Escalation?)
4. Mitigation: Control or defense strategies

Example

"As a malicious user, I want to exploit insecure object references
→ to view another user's LLM interaction history
This results in unauthorized data disclosure
Mitigate: Enforce RBAC and sanitize session references"



Hands-On

Writing Abuse Stories

TOOLS FOR AGILE THREAT MODELING

PyTM

- Overview: A Python-based, open-source tool for automated threat modeling.
- Features: Converts code into a threat model, making it ideal for DevOps and CI/CD pipelines.
- Example Use: Automated threat modeling for cloud infrastructure, identifying potential misconfigurations or unprotected endpoints.

Threat Dragon

- Overview: A visual, drag-and-drop tool for creating threat models.
- Features: Open-source and user-friendly, it’s ideal for smaller teams and projects needing visual representations.
- Example Use: Visualizing attack paths in an application before launch, enabling security-focused design.

Hands-On

PyTM

Hands-On

Threat Dragon

Hands-On

Threat Modeling – GenAI Agile Project

Use the AskBot system diagram and JIRA Ticket (provided)

- Identify Trust Zones, LLM interfaces, Plugins/Tools, and Sensitive Data Flows
- Apply STRIDE + OWASP LLM Top 10 to each component
- Write at least 2 Abuse Stories for the high-risk flows
- Suggest 1 mitigation per identified abuse
- Work in groups of 3–4.
- A mentor will support each table.

Completion of Part 1

Stretch. Hydrate. Breathe.

Part 2 starts in **10 minutes!**

Get your notebooks ready — we'll be building.