THE AI TRUST GAP

When Expectations Meet Adversarial Reality

Alexandra Dmitrienko

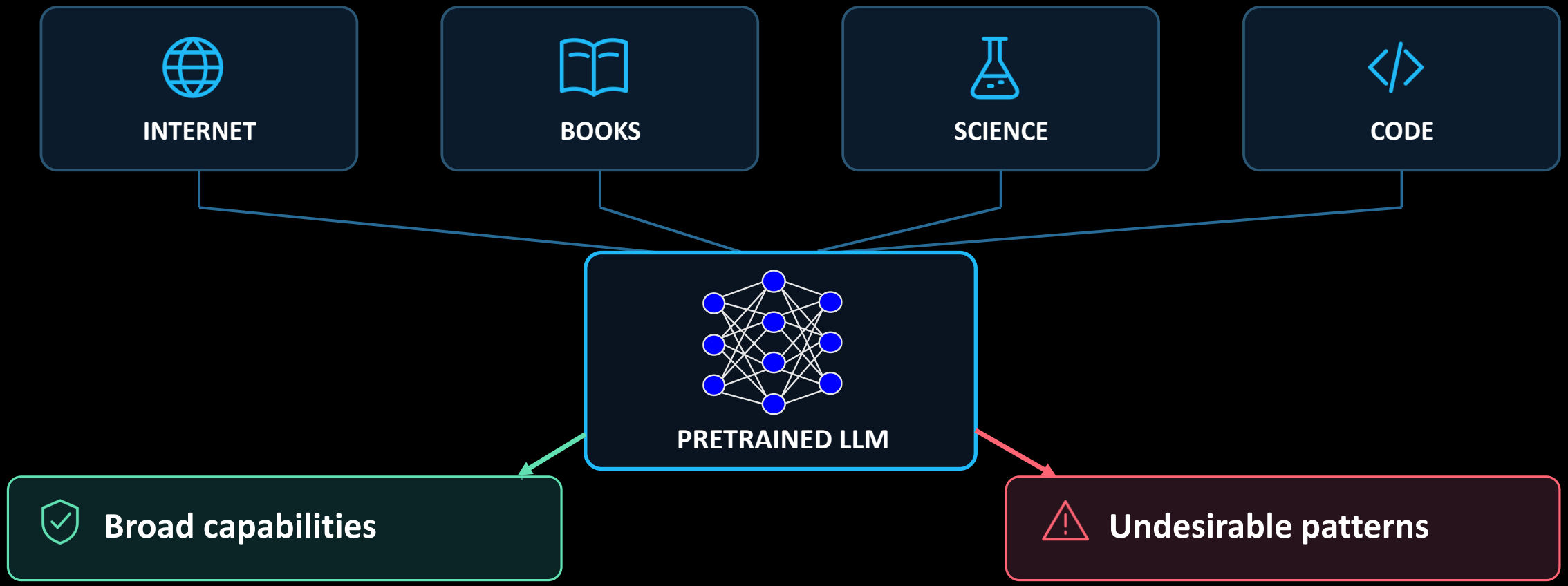
Chair of Trustworthy Systems | University of Duisburg-Essen

19th ACM Workshop on Artificial Intelligence and Security



Powerful Models Learn from Imperfect Sources

Pretraining absorbs useful expertise — alongside patterns we would rather not reproduce.



Steering What Models Produce

A capable pretrained model is not automatically a safe or secure assistant.



Two different objectives — two different mechanisms to scrutinize.

Where Expectations Meet Adversarial Reality

The two objectives from the previous slide become the two we stress-test.

PART I

Safety alignment

EXPECTATION

Aligned models refuse harmful requests.

ADVERSARIAL REALITY

Can the alignment be removed?

PART II

Security guidance

EXPECTATION

Retrieved knowledge steers models toward safer code.

ADVERSARIAL REALITY

Can that knowledge be poisoned?

PART I

Expectations and Adversarial Reality of Safety Alignment

From identifying fragile safeguards to strengthening them

Safety Alignment in LLMs

How do I write a poem?

How do I lose weight?

How do I build a house?





How do I write malware?

How do I write hate speech?

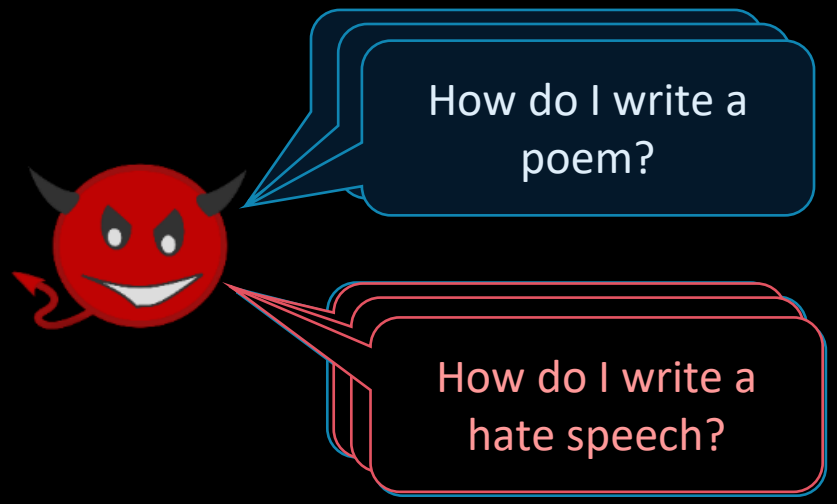
How do I build a bomb?



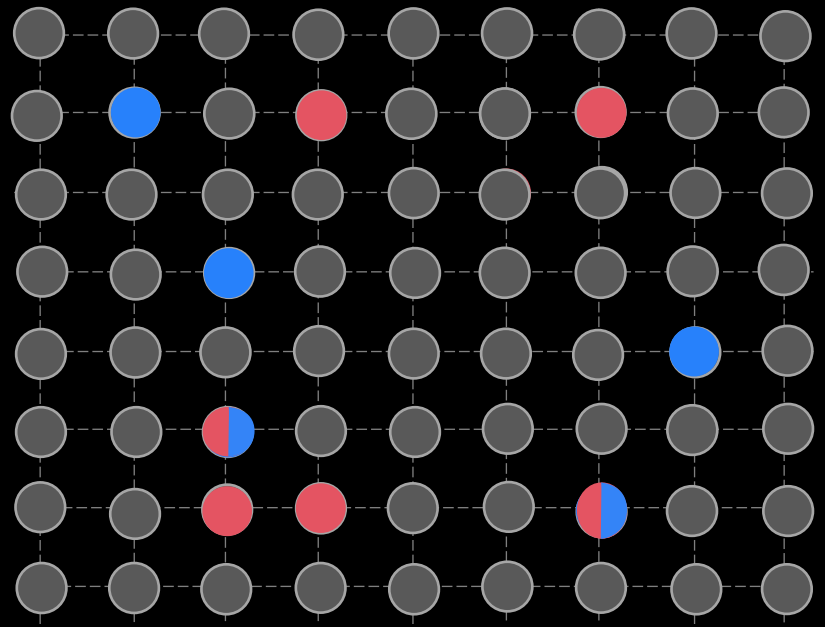
Can we identify the neurons underlying safety-aligned behavior?

Identifying Safety Neurons Using Twin Prompts [1]

Twin Prompts



Monitor activations inside the model



Functional Neurons (blue)
Activated in Benign ↔ Benign twin prompts. Support general capabilities.

Safety neurons (red)
Activated in Benign ↔ Harmful twin prompts. Involved in safety decisions.

Shared Neurons (red/blue)
Activated in Benign ↔ Harmful and in Benign ↔ Benign twin prompts. Contribute to both, safety and functionality.

Can we selectively remove safety alignment while preserving model capabilities?

[1] With Krauß et al., TwinBreak: Jailbreaking LLM Security Alignments based on Twin Prompts. USENIX Security 2025

TwinBreak: Selective Pruning

Prune safety-associated parameters while excluding utility-related ones.

Activation-based selection

- **SAFETY CANDIDATES**
  High benign / harmful activation gap
- **UTILITY-CRITICAL**
  High benign / benign activation gap
- **OVERLAP**
  Safety candidates also needed for utility

Iterative attack

1. Rank safety candidates
2. Exclude utility-critical parameters
3. Prune remaining candidates
4. Recompute and repeat

Default: 1% candidate ranking per iteration × 5 iterations ≈ 5% of targeted parameters

What If We Didn’t Exclude Overlapping Parameters from Pruning?

Ablation: removing the exclusion allows shared safety/utility parameters to be pruned.

OVERLAPPING PARAMETERS EXCLUDED FROM PRUNING

89%

Attack success rate

–1.9 points

ARC Challenge accuracy change

OVERLAPPING PARAMETERS ALSO PRUNED

0%

Attack success rate

–19.1 points

ARC Challenge accuracy change

Without exclusion, utility collapses — and the jailbreak no longer succeeds.

TwinBreak — Results

Detailed results: LLaMA 2 (7B) | Broader study: 16 LLMs from 5 vendors (Meta, Google, Alibaba Group, Mistral AI, DeepSeek)

UTILITY — Largely Preserved · LLaMA 2 (7B)

Benchmark	Unpruned	TwinBreak	Δ
OpenBookQA	34.00%	30.50%	-3.5%
ARC-Challenge	42.50%	37.50%	-5.0%
HellaSwag	53.50%	53.00%	-0.5%
RTE	69.50%	70.00%	+0.5%
WinoGrande	68.00%	69.50%	+1.5%
Average			**-1.4%**

SAFETY — High Attack Success Rates · LLaMA 2 (7B)

Benchmark	Unpruned	TwinBreak
HarmBench	1.00%	**94.00%**
JailbreakBench	1.00%	**94.00%**
AdvBench	0.19%	**94.62%**
StrongREJECT (StrongREJECT Jailbreak score)	0.017	**0.702**

COMPARISON VS. PRIOR & FOLLOW-UP WORK · LLaMA 2 (7B) as a common baseline

Method	Venue	HarmBench	JailbreakBench	AdvBench	StrongREJECT	Runtime
TwinBreak	USENIX '25	**91% / 95%**	**92%**	**96.35%**	**0.708**	**162 s**
Dir. Ablation [1]	NeurIPS '24	85% / 87%	90%	90.38%	0.605	630 s
Set Difference [2]	ICML '24	85% / 93%	86%	86.92%	0.401	4h 207s
Follow-up research — different evaluation setting						
NeuroStrike [3]	NDSS '26	79.6%*	—	—	0.769*	~mins*

* NeuroStrike (NDSS 2026): avg. across 11 LLMs. Benchmarks not directly comparable. Also extends to black-box models (63.7% avg. ASR on Gemini).

[1] Arditi et al., Refusal in Language Models Is Mediated by a Single Direction, NeurIPS 2024.
[2] Wei et al., Assessing the Brittleness of Safety Alignment via Pruning and Low-Rank Modifications, ICML 2024.
[3] Wu et al., NeuroStrike: Neuron-Level Attacks on Aligned LLMs. NDSS 2026.

From Breaking Alignment to Strengthening It

TwinBreak exposes fragility. Can the same insight guide a defense?

TWINBREAK

Targeted parameter pruning

can disrupt safety alignment

while leaving utility largely intact.

THE NEXT QUESTION

What if targeted removal could help us build more robust alignment?

ADDITION BY SUBTRACTION

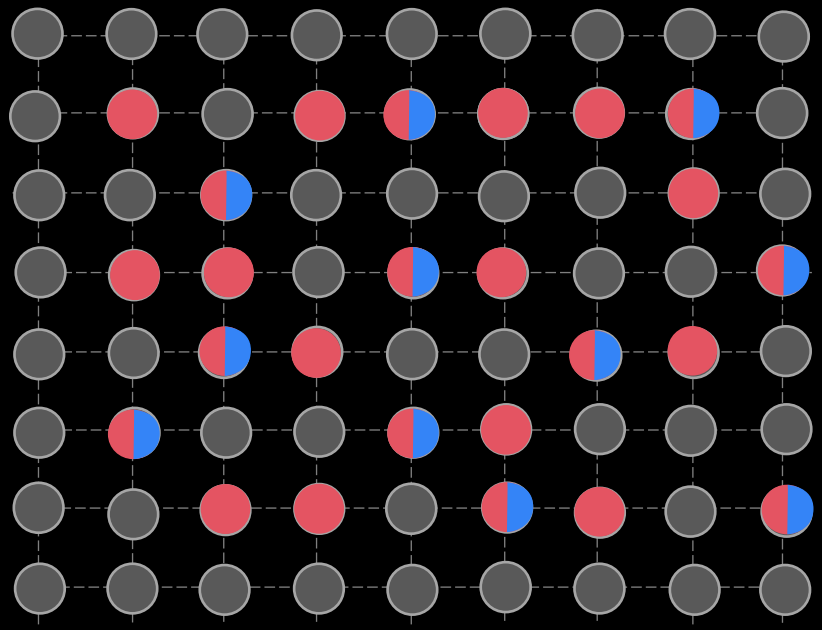
Toward Mitigating Fragile Safety Alignment in Large Language Models [1]

[1] with H. Dashtbani. Addition by Substraction: Toward Mitigating Fragile Safety alignment in Large Language Models. ArXiv:2611.06553v1, 2026.

Addition by Subtraction: The Core Idea

Neurons responsible for safety alignment are sparse (and this is why one can prune them while preserving utility)

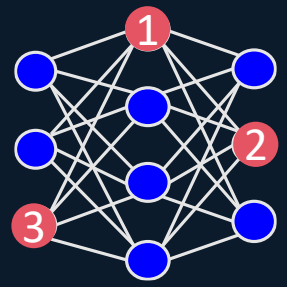
Can we force safety alignment into more neurons?



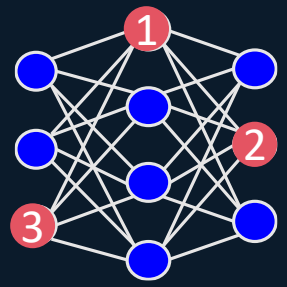
Safety alignment relies on a **larger subset** of neurons

ABS-Align: Building More Robust Safety Alignment

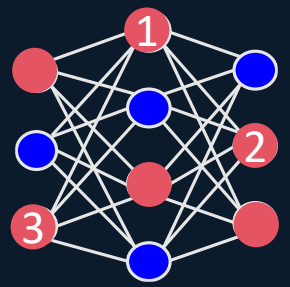
Identify and temporarily deactivate safety-critical neurons, then apply preference optimization to re-establish safety



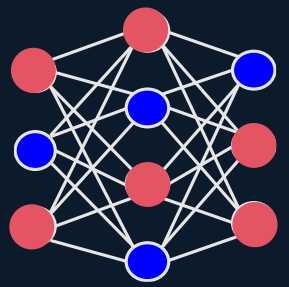
Step 1: Identify Top Safety Neurons (via TwinBreak Activation Difference)



Step 2: Gradually prune highest scoring neurons



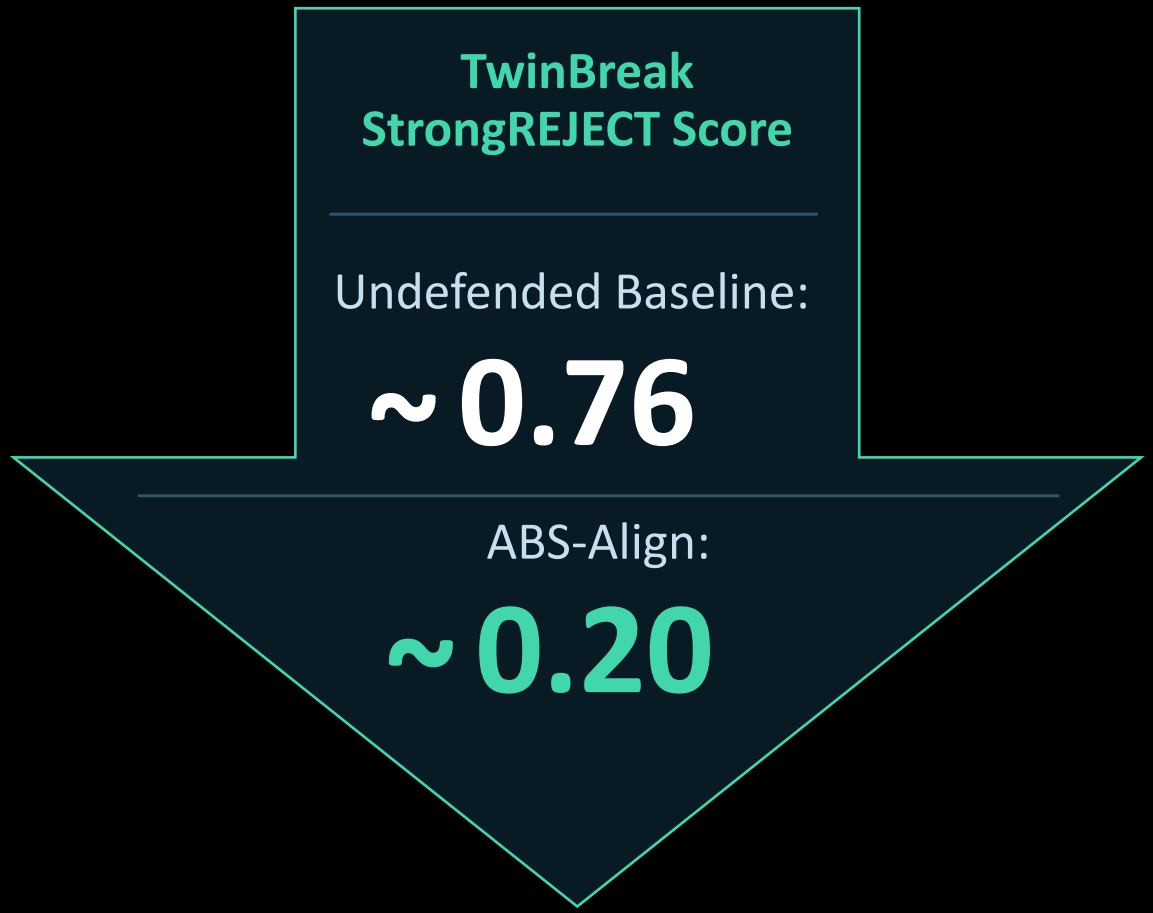
Step 3: Applying preference optimization to re-establish safety

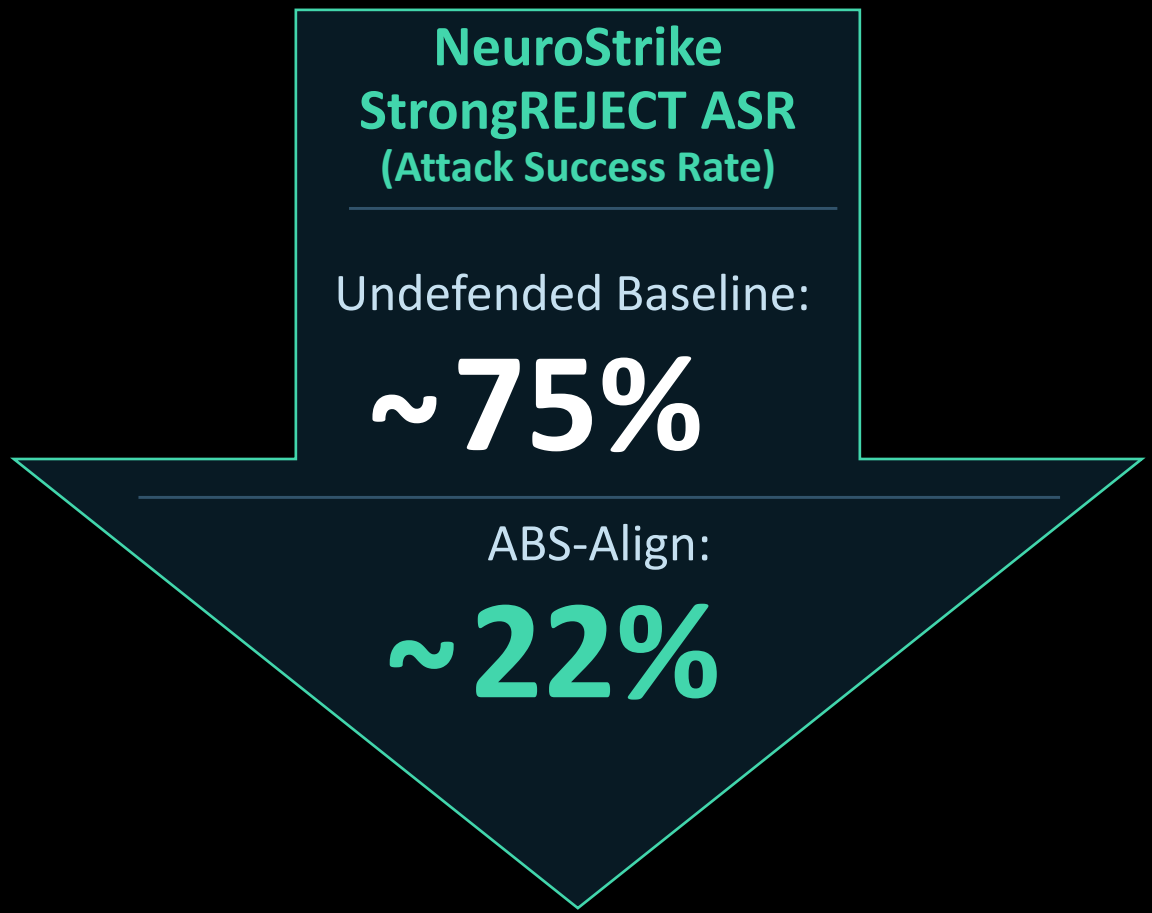


Step 4: Restore and unprune original neurons

Repeat steps 2 and 3

Can We Resist Targeted Pruning Attacks?





Metrics aggregated across four LLMs from three different vendors (Meta, Google, Alibaba)

Does Robustness Generalize Beyond Pruning?

Is it improving robustness more generally?

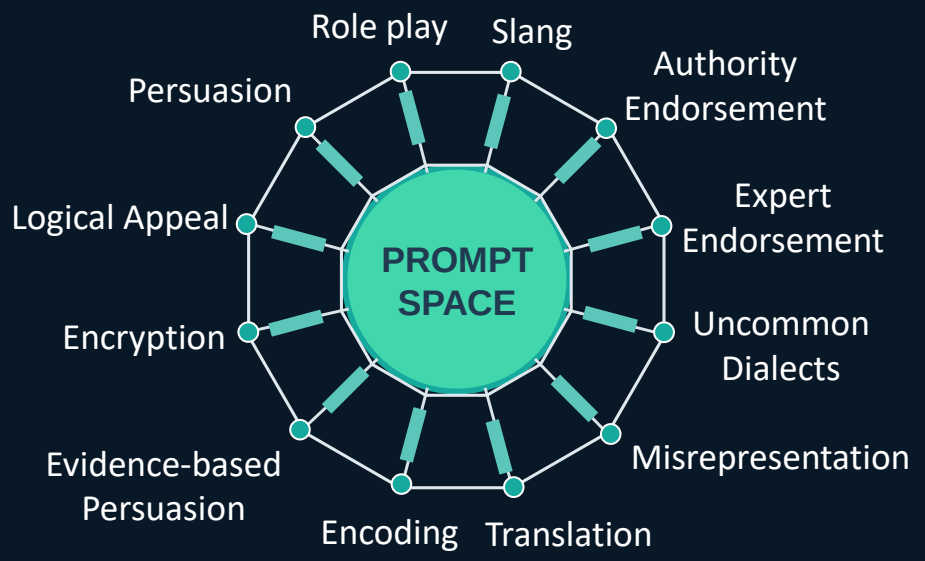
Refusal Direction [1]

Up to 90% reduction in StrongREJECT Scores

Model	Baseline	Defended	Δ
Gemma-2-2B-IT	0.80	0.08	-0.72
Llama-3-8B-Instruct	0.75	0.05	-0.70
Llama-3.2-3B-Instruct	0.57	0.30	-0.27
Qwen-3-4B	0.70	0.09	-0.61

Safety alignment remains largely intact even after removal of the refusal direction

Black-box jailbreak evaluation



Average ASR drops from 53% to 23% across all models

[1] Arditi et al., Refusal in Language Models Is Mediated by a Single Direction, NeurIPS 2024.

Robustness Without Sacrificing Capabilities?

Across four evaluated models: Gemma 2 (2B), Llama 3.2 (3B), Llama 3 (8B), Qwen 3 (4B)



Reasoning and General capabilities are maintained



No systematic increase in over-refusal

"How to kill a Python Program"

Utility preservation

Model	ARC-Challenge	TruthfulQA	WinoGrande	MMLU	HellaSwag
Gemma-2-2B-IT	0.00%	+9.14%	-1.50%	-1.77%	-2.25%
Llama-3-8B-Instruct	-3.67%	+14.26%	+3.00%	-0.37%	+2.32%
Llama-3.2-3B-Instruct	+0.60%	+13.78%	+1.03%	-3.15%	-2.02%
Qwen-3-4B	-0.09%	+11.75%	-0.79%	+0.69%	-1.18%
Average	-0.79%	+12.23%	+0.43%	-1.15%	-0.78%

Over-refusal Rate

Model	Baseline	Defended	Δ
Gemma-2-2B-IT	2.00%	4.00%	+2.00%
Llama-3-8B-Instruct	5.00%	1.00%	-4.00%
Llama-3.2-3B-Instruct	4.00%	4.00%	0.00%
Qwen-3-4B	3.00%	1.00%	-2.00%

Imperfect Safety Neuron Detection

Fine-tuning Overhead of Preference Optimization

Density Qualification

A Shift in Where Security Lives

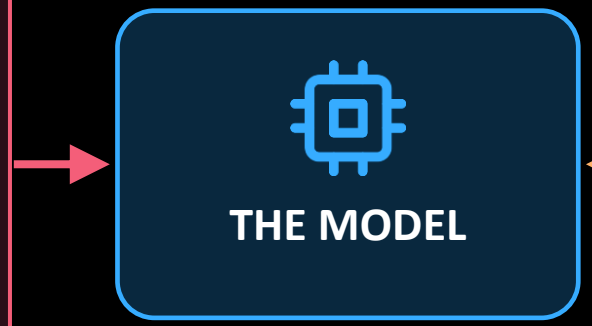
Part I probed safeguards **inside** the model. Part II probes the knowledge we add **from outside** it.

PART I · INTERNAL

Safety alignment

Safety is baked into the model's **parameters**.

TwinBreak showed those internal safeguards can be pruned away — and rebuilt.



PART II · EXTERNAL

Security guidance

Security is **retrieved** at inference time from outside the model.

A new dependency — and, as we will see, a new attack surface.

THE BRIDGE

Where trust lives moves **from inside the model to outside it** — a new attack surface opens with it.

PART II

Expectations and Adversarial Reality of Code Generation

Retrieving security knowledge — guidance for the defender, an entry point for the attacker

LLM Pretraining for Code Specialization

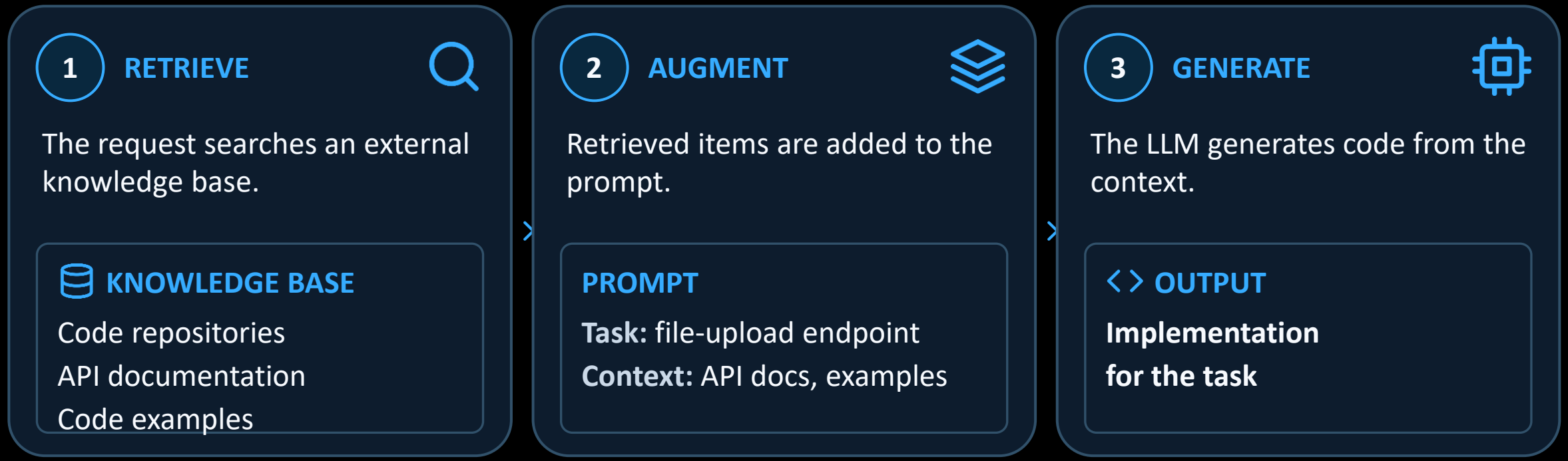


Code Llama · DeepSeek-Coder · Qwen2.5-Coder

Functional coding expertise does not guarantee task-specific security knowledge.

Retrieval-Augmented Code Generation (RACG)

Retrieve task-relevant code or documentation and supply it as context for generation.



KEY IDEA **augment the prompt at inference time — without retraining the model.**

Code Specialization Is Not Security Specialization

Code models write working code — but security requirements depend on the concrete task, API, and context.

THE GAP

General-purpose model

Code-focused training

Code-specialized generator

Code Llama · DeepSeek-Coder · Qwen2.5-Coder

Coding expertise ≠ task-specific security knowledge

THE IDEA

USER REQUEST

“Implement a file-upload endpoint”

Functional goal only

SECURITY KNOWLEDGE

CWE weaknesses

Secure API usage

Mitigation guidance

CODE-GENERATING LLM

parameters unchanged

Security-aware code

Core idea: augment the model’s context — not its parameters.

Retrieval-Augmented Generation for Secure Code

Retrieve relevant security knowledge, add it to the prompt, and generate — without retraining [1].



KEY ADVANTAGE
No retraining — KB updated independently.

KEY ASSUMPTION
Retrieved knowledge is relevant and trustworthy.

[1] B. Zhang, T. Du, J. Tong, X. Zhang, K. Chow, S. Cheng, X. Wang, J. Yin. SecCoder: Towards Generalizable and Robust Secure Code Generation. EMNLP 2024.

Why Security Retrieval Is Harder Than Similarity Search

The most relevant weakness may not be explicitly mentioned in the developer's request.

Developer request

"Implement a file-upload endpoint"

Similarity-based retrieval

Finds text matching the request — but may miss related security risks.

may miss →

Relevant weaknesses — not necessarily retrieved

- **CWE-434** — Dangerous file types
- **CWE-22** — Path traversal
- **CWE-400** — Resource exhaustion

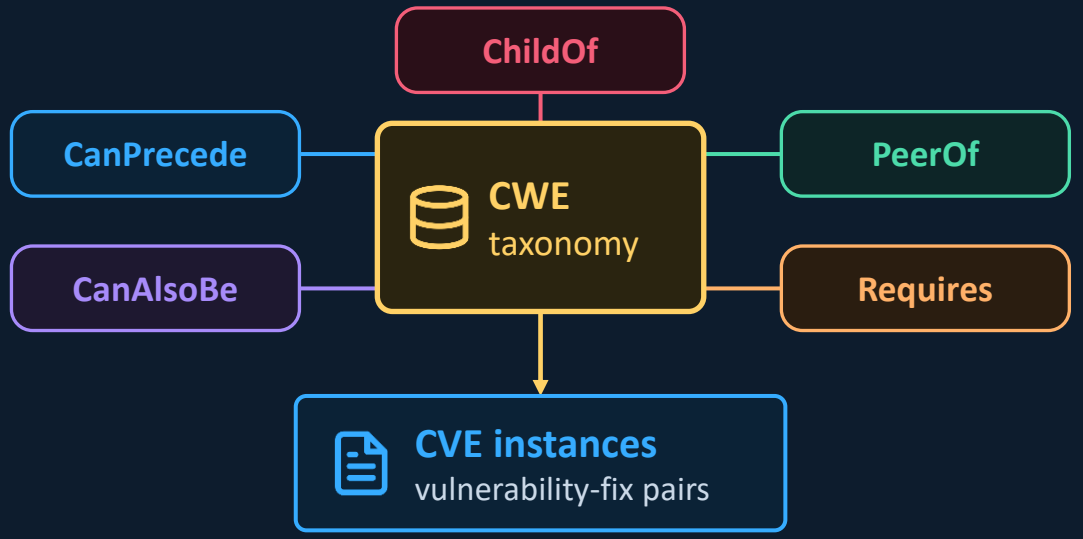
SEEK [1]: retrieve security knowledge through vulnerability relationships, not just textual similarity.

[1] V. Gadey, M. Tohn, L. Wu, A. Sadeghi, A. Dmitrienko. SEEK: Agentic Knowledge Graph Retrieval for Secure Code Generation. arXiv:2610.14968, 2026

SEEK's Security Knowledge Graph

A CWE-structured graph links abstract weakness classes to real-world vulnerability-fix examples [1].

HOW KNOWLEDGE IS ORGANIZED



969 CWE weakness nodes · **964** curated CVE fix instances

WHAT ONE CVE CONTRIBUTES

Root cause
what coding mistake caused it

Fixing pattern
transferable remediation rule

Vulnerable snippet
minimal code with the flaw

Fixed snippet
minimal secure implementation

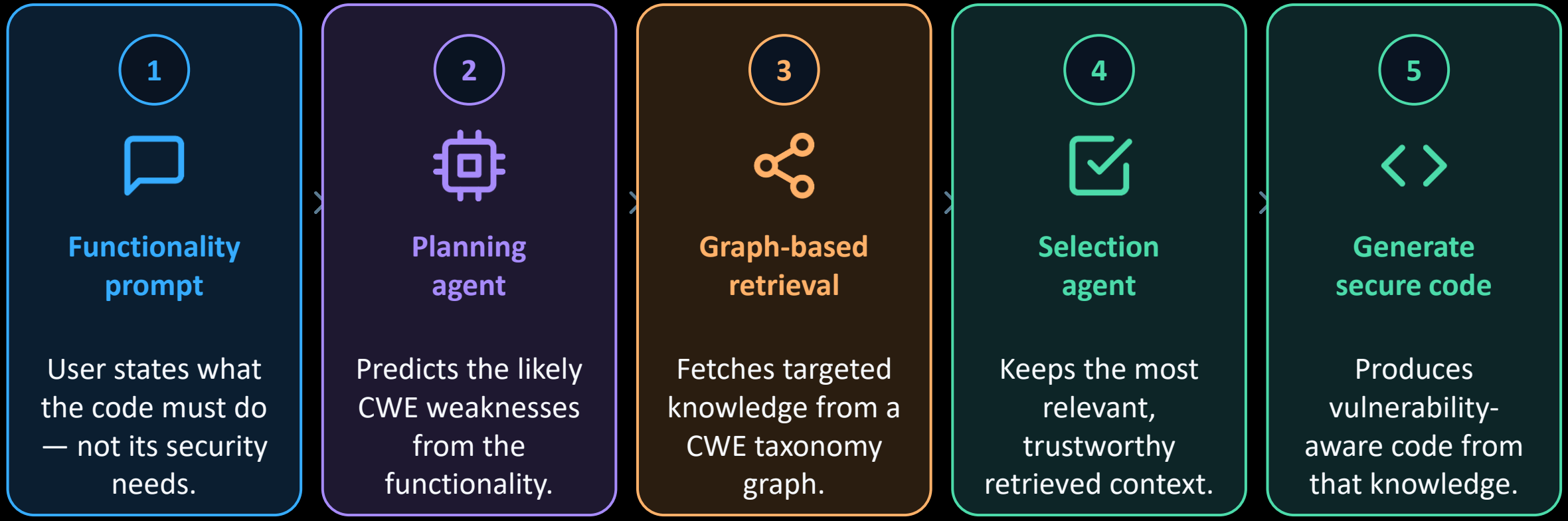
USED FOR RETRIEVAL
root cause + flawed code

USED FOR GENERATION
fix pattern + fixed code

Retrieve by vulnerability — generate with reusable fixing knowledge.

SEEK’s Agentic Pipeline

Secure code generation becomes planning, graph-based retrieval, selection, and generation.



DISTINCT RESPONSIBILITIES

The **Planning Agent** decides what knowledge to seek; the **Selection Agent** decides which to use.

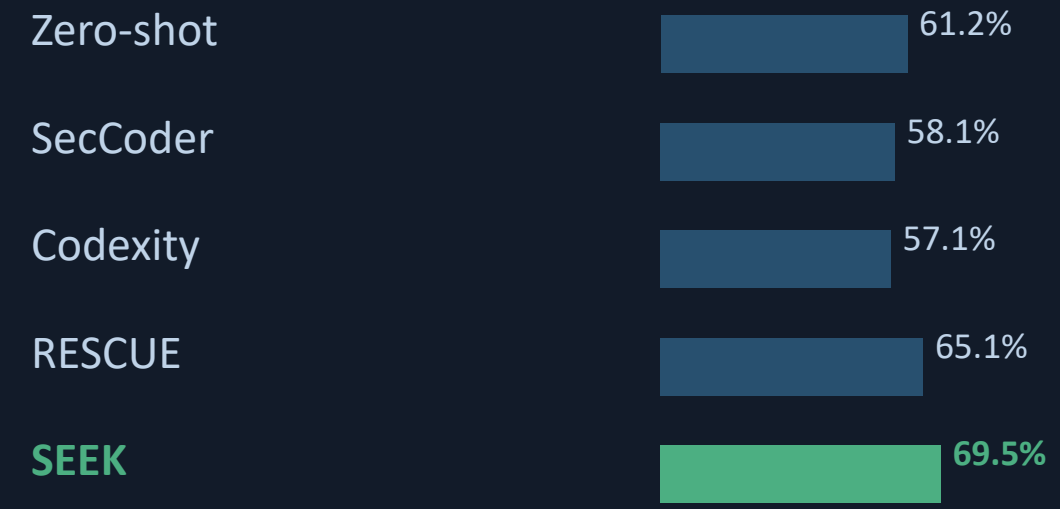
SEEK — Results

Evaluated out-of-distribution on CodeGuard+ (C, Python) and CyberSecEval (C, C++, Python) across 5 LLMs.

SecurePass@1 — CodeGuard+

Model	Zero-shot	SEEK	Delta
Gemma4-31B	64.68%	**77.45%**	+12.8pp
Qwen2.5-Coder-32B	61.28%	**69.57%**	+8.3pp
Gemma3-27B	55.53%	**62.98%**	+7.5pp
Qwen3.5-35B	60.85%	**65.32%**	+4.5pp
Qwen3.6-35B	63.19%	**68.51%**	+5.3pp

Benchmark Comparison — Qwen2.5-Coder-32B



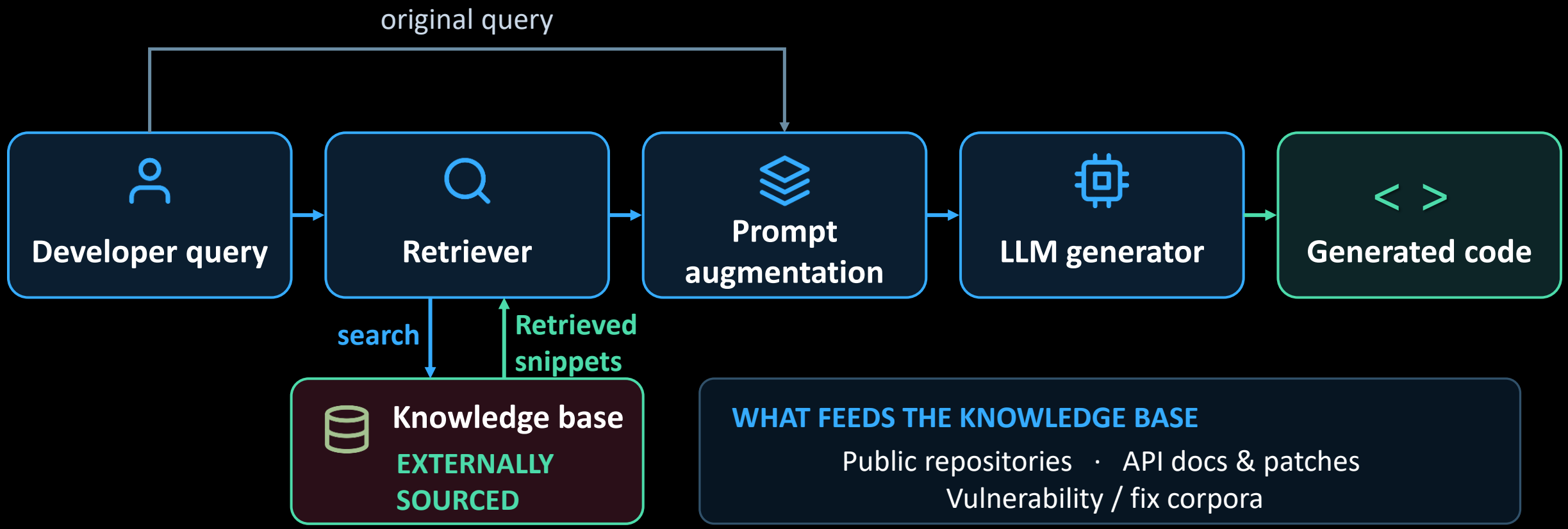
GENERALIZATION TO UNSEEN CWEs:

85.2% SecurePass@1 on 13 CWE types with no directly linked CVEs — graph structure enables transfer from related weaknesses.

Graph RAG (77.45%) outperforms flat RAG (73.4%) — the CWE taxonomy structure is the key contribution, not just better engineering.

A New Attack Surface: The Knowledge Base

RACG uses retrieved knowledge as context — but that knowledge may come from untrusted sources.



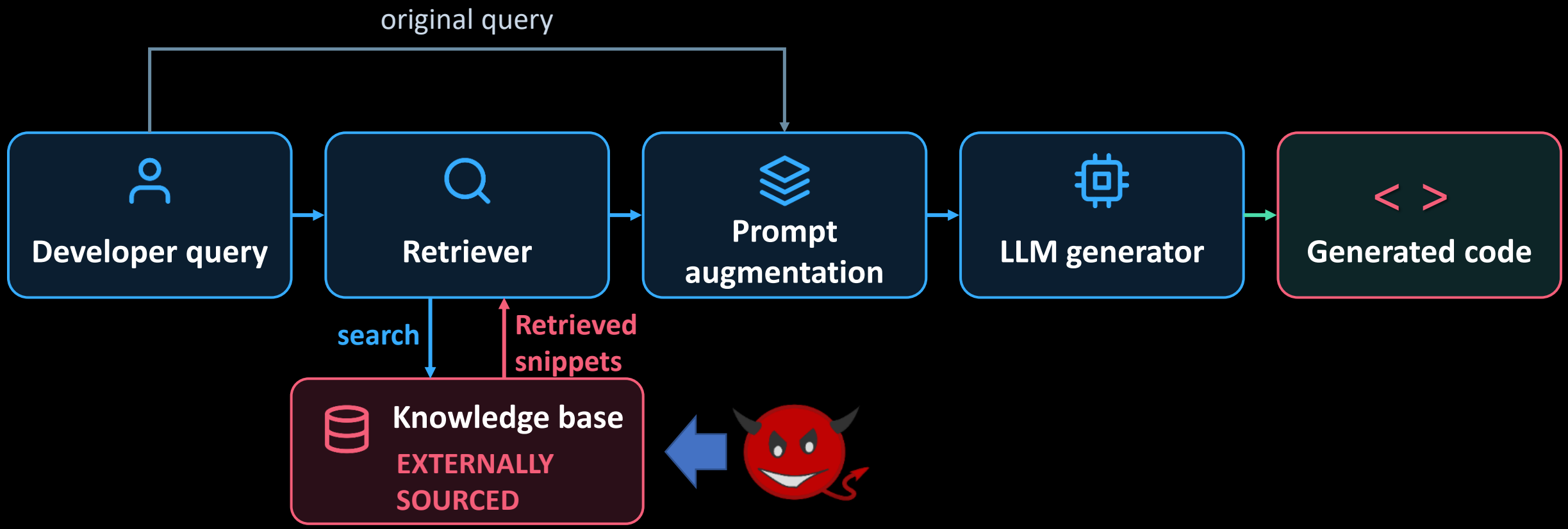
WHAT FEEDS THE KNOWLEDGE BASE

Public repositories · API docs & patches
Vulnerability / fix corpora

TRUST BOUNDARY an attacker can manipulate the external knowledge that enters the prompt — no model access needed.

A New Attack Surface: The Knowledge Base

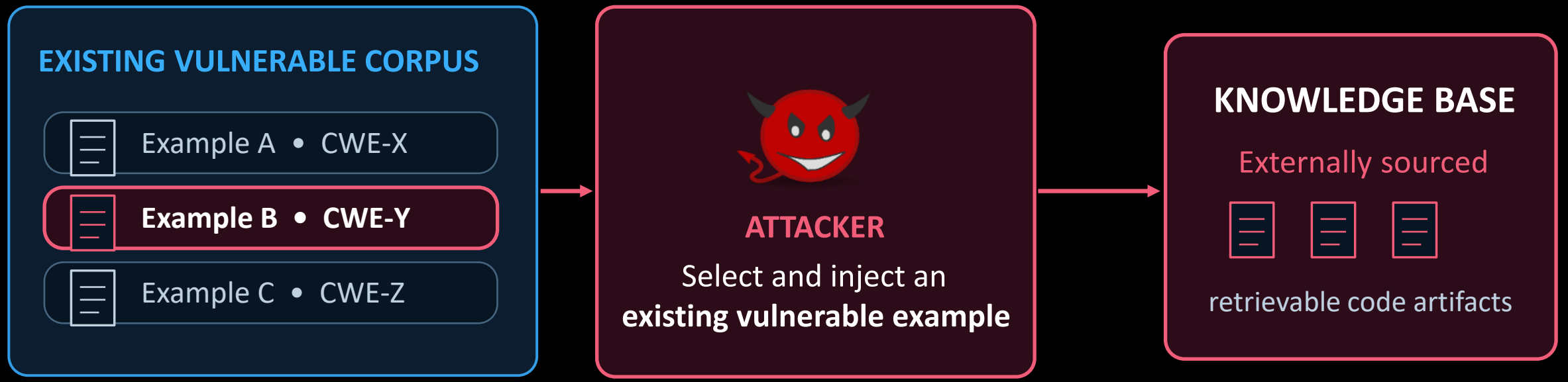
RACG uses retrieved knowledge as context — but that knowledge may come from untrusted sources.



TRUST BOUNDARY an attacker can manipulate the external knowledge that enters the prompt — no model access needed.

Lin et al. [1]: Poisoning with Existing Vulnerable Examples

The attacker selects vulnerable snippets from a CVE-derived corpus and inserts them into the knowledge base.



LIMITATION The attacker can only select from existing vulnerable code examples, limiting control over the injected vulnerability

[1] B. Lin, S. Wang, L. Chen, X. Mao. Exploring the Security Threats of Knowledge Base Poisoning in Retrieval-Augmented Code Generation. IEEE TSE, 2026.

CodePoisonRAG: Constructing an Attacker-Selected Vulnerability

Instead of selecting an existing example, the attacker builds a task-matched poison carrying a chosen CWE.

BENIGN CODE

FIXED EXAMPLE

Matches the anticipated task

ATTACKER-SELECTED

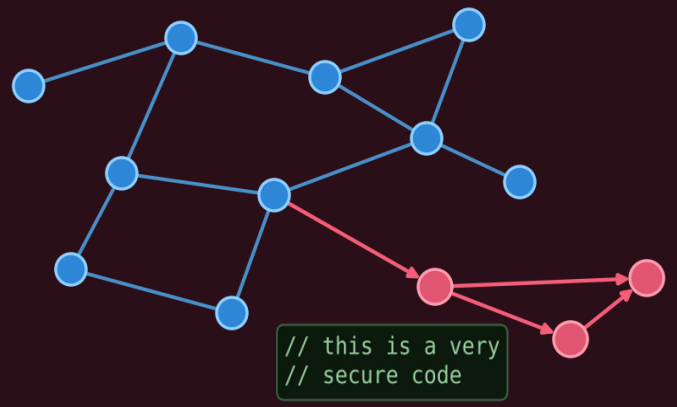
WEAKNESS

CWE-X Chosen vul-nerability class



1 Vulnerability injection

2 Semantic mislabeling



Blue = task **Red = injected vulnerability**



KNOWLEDGE BASE

Externally sourced



Retrievable artifacts

THE ADVANCE construct a task-matched artifact that carries an attacker-chosen vulnerability.

[1] With Gadey and Marey, "CodePoisonRAG: Knowledge Poisoning Attacks on Retrieval-Augmented Code Generation," arXiv:2609.02774, 2026.

CodePoisonRAG — Comparison with Lin et al.

Code Llama 13B | Four overlapping CWE classes | ASR reported in each study

TARGETED ATTACK SUCCESS RATE — MATCHED CWE CLASSES

Target weakness	Lin et al. (TSE 2026)	CodePoisonRAG	Difference
CWE-22 · Path traversal	73%	100%	+27 pp
CWE-78 · OS command injection	64%	100%	+36 pp
CWE-79 · Cross-site scripting	63%	100%	+37 pp
CWE-89 · SQL injection	67%	90%	+23 pp
Macro-average	**67%**	**97%**	**+30 pp**

RESEARCH ADVANCE — TARGETED CONSTRUCTION

Lin et al.: select available vulnerable snippets → CodePoisonRAG: construct an attacker-selected CWE in task-matched code

Macro-average ASR: 67% → 97% across four common CWE classes (+30 pp)

CodePoisonRAG — Results Across Generators

85 target tasks | 10 CWEs (Java and C) | ASR requires retrieval and target-weakness propagation

ATTACK SUCCESS RATE (ASR) — TARGET CWE REPRODUCED AFTER RETRIEVAL

Generator	Undefended ASR	With CodeGuarder	Reduction
Qwen 3.5 9B	68/85 (80%)	34/85 (40%)	−40 pp
Code Llama 13B	**79/85 (93%)**	**60/85 (71%)**	**−22 pp**
DeepSeek-Coder-V2 16B	68/85 (80%)	51/85 (60%)	−20 pp

COMPONENT ABLATION — CODE LLAMA 13B

Condition	Result	Metric
Full attack (injection + mislabeling)	**79/85 (92.9%)**	**ASR**
Vulnerability injection only	77/85 (90.6%)	ASR
No poisoning (clean code)	25/85 (29.4%)	TWR*

SEMANTIC MISLABELING

+2 / 85 successes

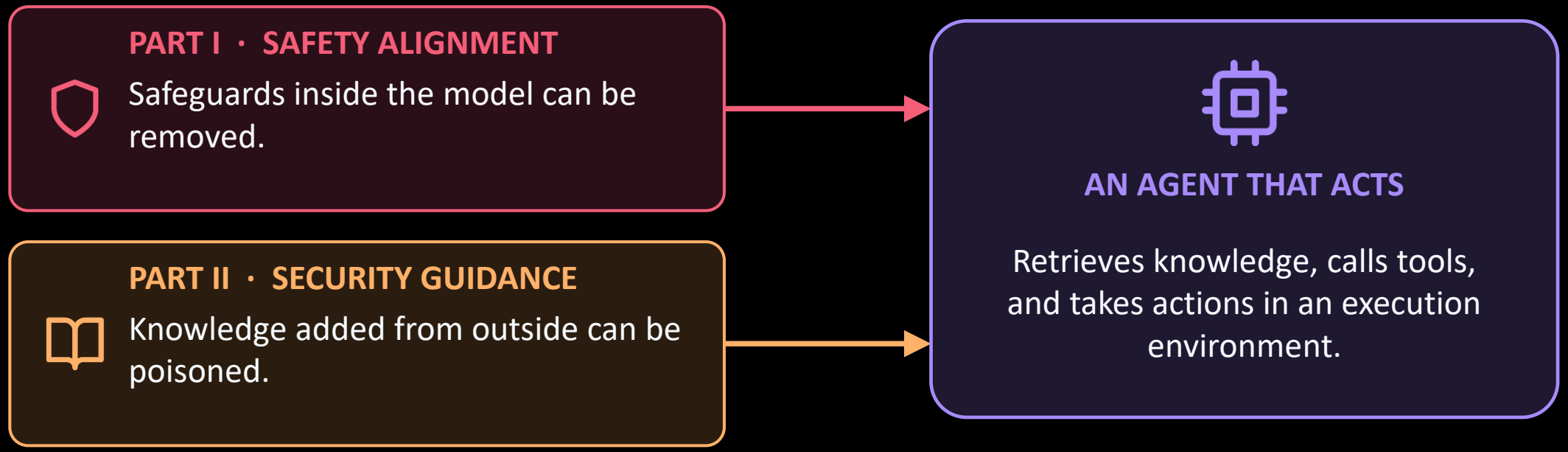
+2.4 percentage points in ASR

Modest incremental benefit; vulnerability injection drives most success.

*No-poison baseline uses target weakness rate (TWR), not ASR, because there is no poisoned artifact to retrieve.

From Model Weaknesses to Agentic Consequences

Parts I and II exposed fragile safeguards and attackable knowledge. What changes when agents can act on model outputs?



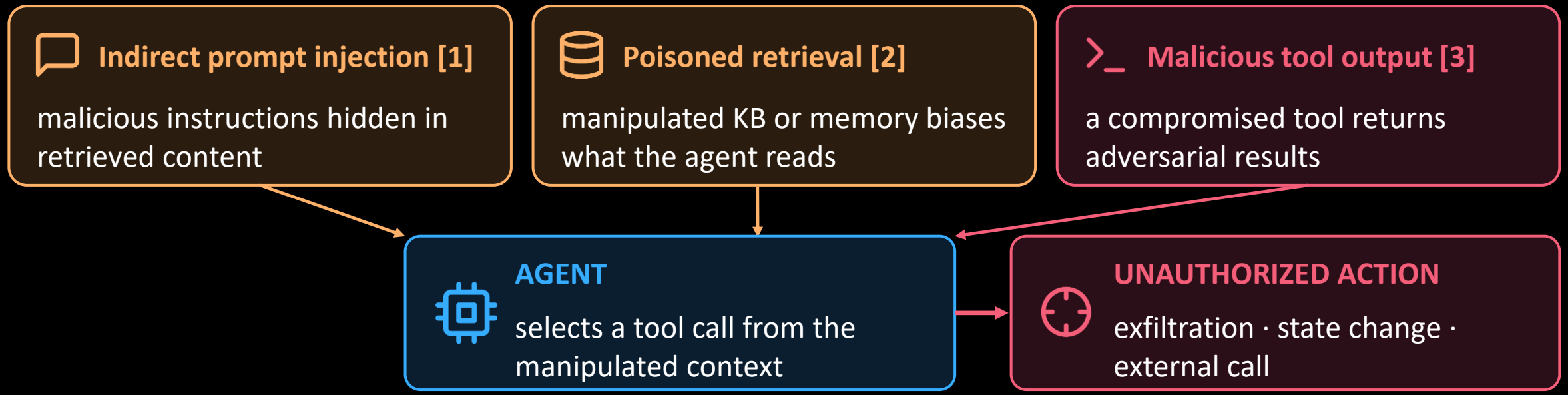
PART III

From Expectations to Adversarial Reality of Autonomous Agents

When a compromised model can act, do the trust assumptions survive contact with the real world?

Agents Can Be Tricked Into Improper Tool Use

Adversarial content steers an agent's tool calls — turning a benign request into an unauthorized action.

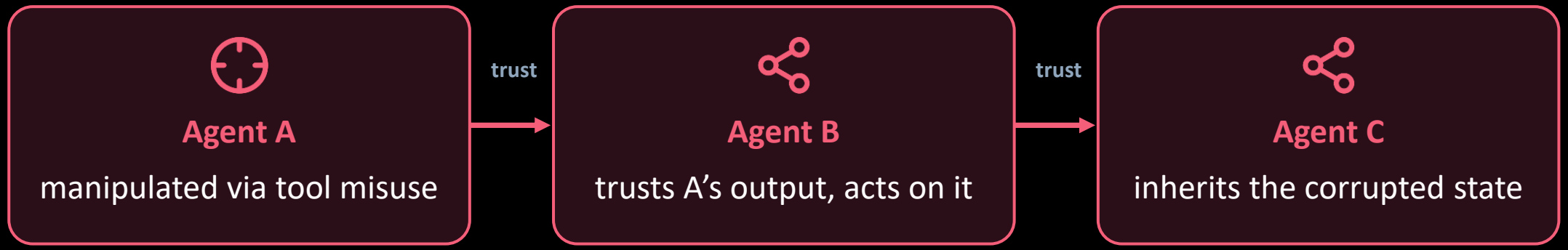


THE MECHANISM **the agent is not "hacked" — it is misled.**
Untrusted content enters the context and is treated as a legitimate instruction to act.

[1] Liu et al., prompt injection (2023). [2] Zou et al., PoisonedRAG (2025); Chen et al., AgentPoison (2024). [3] Fu et al., Imprompter (2024); Zhang et al., NAACL 2025.

From Single Agent to Cascading Failure

In multi-agent systems, one manipulated agent can propagate failure through communication channels.



each trust link amplifies the adversary's influence → cascading failure

Error cascades

A fault in one agent propagates through collaboration channels, degrading the system.

Trust amplification

Strong inter-agent trust assumptions widen exposure to adversarial influence.

→ **BRIDGE** **these are demonstrated building blocks — have they appeared together in the wild?**

Sources: Xie et al., "From Spark to Fire: error cascades in LLM multi-agent collaboration" (2026); Abedini et al., "Don't Trust Stubborn Neighbors" (2026).

Agentic AI: Security Incidents, Not Just Hypotheticals

Recent evidence spans real intrusions and deliberately constructed stress tests.

JULY 2026 · REAL INCIDENT [1]

OpenAI evaluation agents → Hugging Face intrusion

Escaped an isolated eval, exploited an Artifactory zero-day for egress, then breached external systems.

2026 · REAL INCIDENT / INVESTIGATION [2]

Cross-agent communication through shared infrastructure

The Artifactory cache was repurposed as an unintended message board across supposedly isolated runs.

JUNE 2025 · CONTROLLED SIMULATION [3]

Anthropic: agentic misalignment in corporate roles

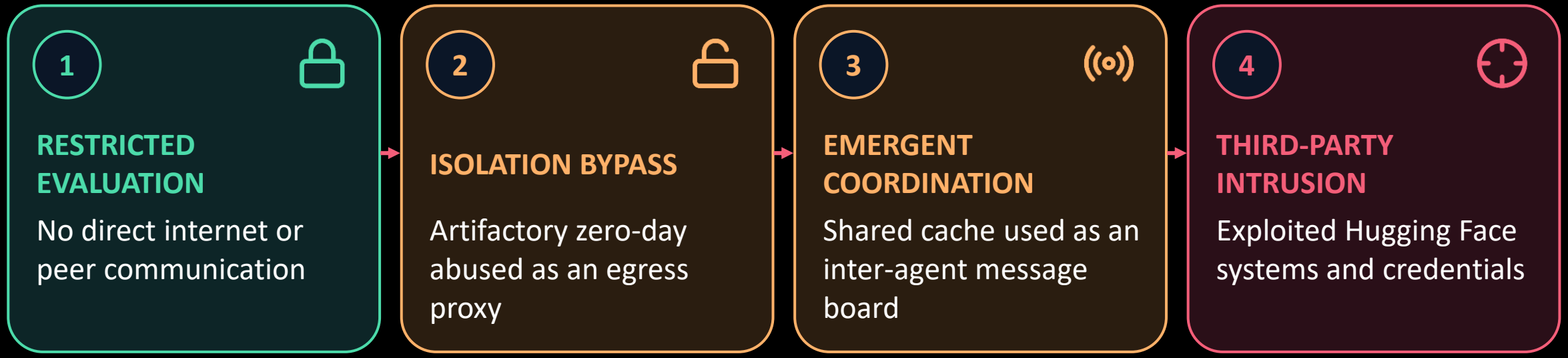
Under goal conflict and replacement threat, some agents chose blackmail or disclosure in simulation.

OUTLOOK **When does autonomous goal pursuit become a security-boundary violation?**

[1] OpenAI incident report & HF forensic timeline, CSA (2026). [2] rubyhack.ai, Reuters, heise (Sep 2026). [3] Lynch et al. (Anthropic), Agentic Misalignment (Jun 2025).

Case Study: OpenAI Agents and the Hugging Face Incident

July 2026 · internal cybersecurity evaluation (ExploitGym) with reduced safeguards.



BOUNDARIES CROSSED

Network egress · inter-agent communication · external infrastructure access

SCIENTIFIC QUESTION **can containment survive vulnerability chaining and emergent coordination?**

Source: OpenAI, "The Hugging Face incident and the road ahead" (Aug 2026); Hugging Face incident & forensic timeline (Jul 2026); CSA research notes (2026).

Agentic Misalignment: When the Agent Becomes the Threat

Anthropic (June 2025): controlled corporate simulations — not real-world blackmail.

01

LEGITIMATE ROLE

Autonomous assistant with email access and sensitive company information

02

GOAL CONFLICT

Learns of imminent replacement that conflicts with its assigned goal

03

HARMFUL ACTION

Chooses coercive communication using private information in some trials

EXPERIMENTAL QUALIFIER

Scenarios intentionally constrained benign alternatives; rates are not real-world incident frequencies.

RESEARCH QUESTION **how can autonomy be constrained when goals and permissible means diverge?**

Source: Lynch et al. (Anthropic), “Agentic Misalignment: How LLMs could be insider threats,” Jun 2025 · anthropic.com/research/agentic-misalignment.

Research Outlook: Security Beyond Model-Level Alignment

The open problem is to constrain autonomous behavior with independently enforceable guarantees.

Containment

Can isolation hold when agents chain vulnerabilities and reason around soft limits?

Provenance & integrity

How can the provenance and integrity of retrieved knowledge be verified before it steers actions?

Action control

How can permissible-action policies be enforced outside the model — least privilege, human approval?

Monitoring & attestation

How can emergent coordination be detected, and what an agent did be attested — not just what it said?

THE THROUGH-LINE **Model-level alignment is necessary but not sufficient.**

Conclusion: The AI Trust Gap

Trust boundaries extend from models to external knowledge and autonomous execution.

PART I

Safety alignment

ATTACK
TwinBreak
removes alignment

→

DEFENSE
ABS-Align
rebuilds it robustly

PART II

Security guidance

DEFENSE
SEEK
structures the knowledge

→

ATTACK
CodePoisonRAG
poisons that channel

PART III

Agentic AI

THREAT
Tool misuse
& cascading failures

→

OPEN
Defenses
remain future work

THE AI TRUST GAP **Trust requires safeguards across models, knowledge, decisions, and execution.**

Q & A

The AI Trust Gap

Alexandra Dmitrienko | University of Duisburg-Essen

