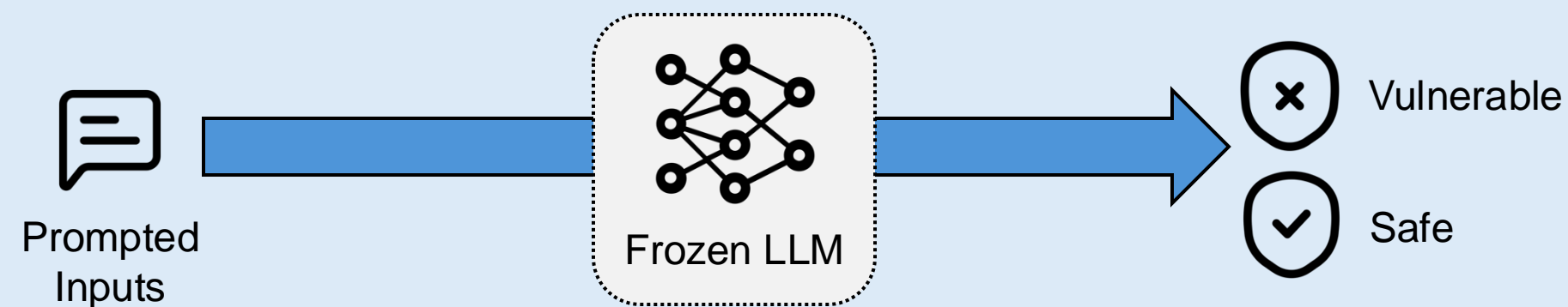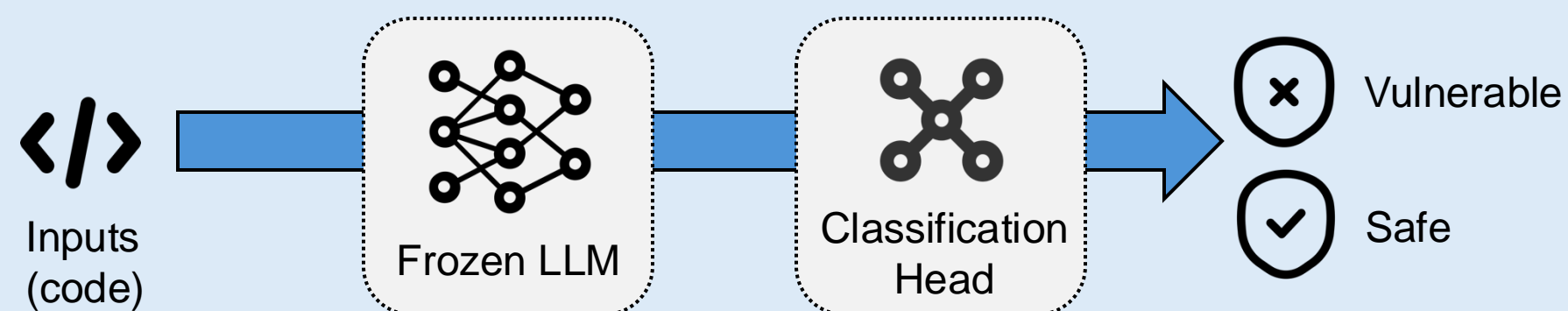# LLM-Based Methods

① Zero-Shot Classification

User: Answer ONLY with 'Yes' or 'No' and provide no additional text: Does the following JavaScript code snippet contain any exploitable Arbitrary Code Execution (ACE) or Arbitrary Command Injection (ACI) vulnerabilities?
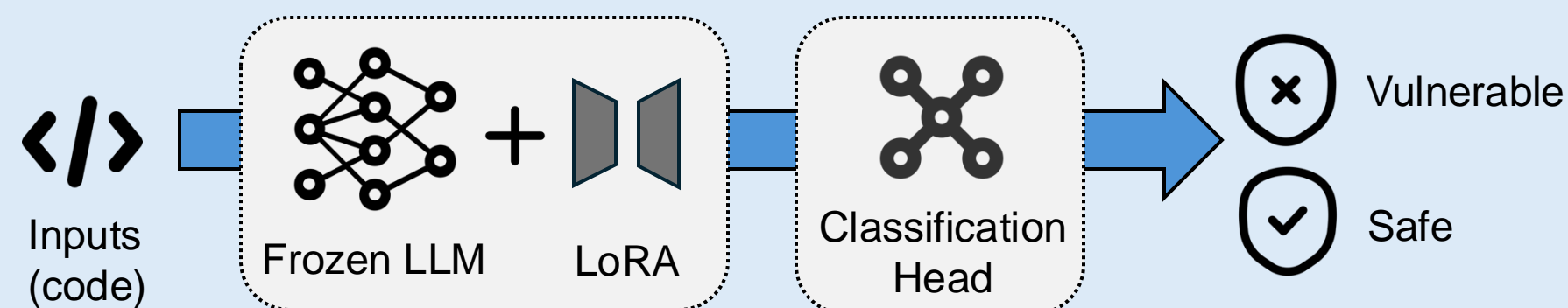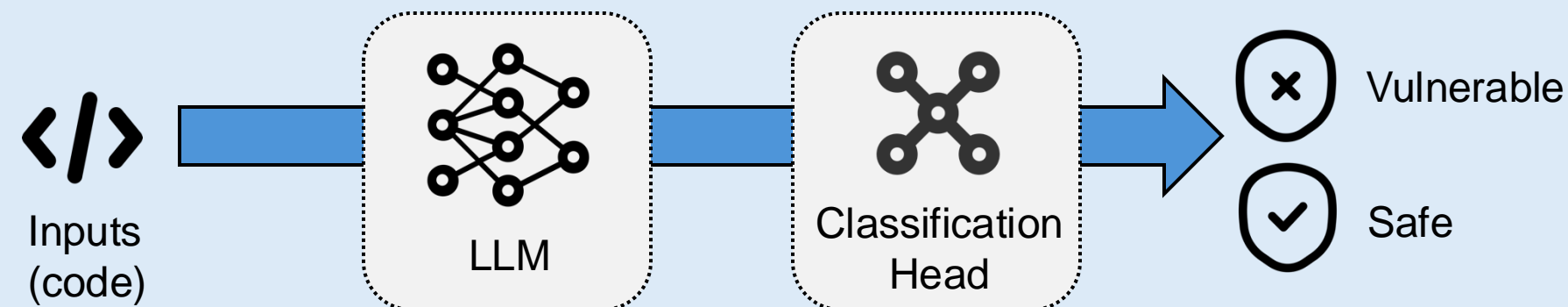
{code snippet here}

Prompted Inputs → Frozen LLM → ✕ Vulnerable / ✓ Safe

② Fine-Tuning on Classification Head

Inputs (code) → Frozen LLM → Classification Head → ✕ Vulnerable / ✓ Safe

③ Fine-Tuning (LoRA)

Inputs (code) → Frozen LLM + LoRA → Classification Head → ✕ Vulnerable / ✓ Safe

④ Full Fine-Tuning

Inputs (code) → LLM → Classification Head → ✕ Vulnerable / ✓ Safe

Program Analysis Methods (Including ML-Enhanced Versions)

⑤ FAST

Inputs (code) → Abstract Syntax Tree Generation → Path Generation → Exploit Generation → ✕ Vulnerable / ✓ Safe

⑥ NODEMEDIC

Inputs (code) → Driver Generation → Provenance Tracking → Exploit Generation → ✕ Vulnerable / ✓ Safe
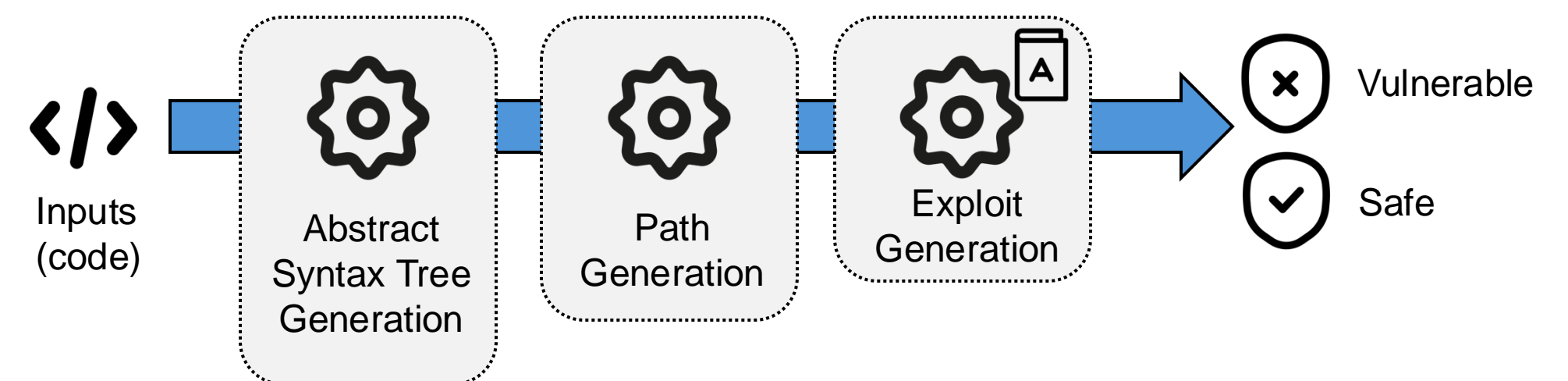
Vulnerable Path Detection (NODEMEDIC)

⑦ NODEMEDIC-GNN

Inputs (code) → NODEMEDIC → GNN → Classification Head → ✕ Vulnerable / ✓ Safe

⑧ NODEMEDIC-ML

Inputs (code) → NODEMEDIC → Pooling → ML Classifiers → ✕ Vulnerable / ✓ Safe
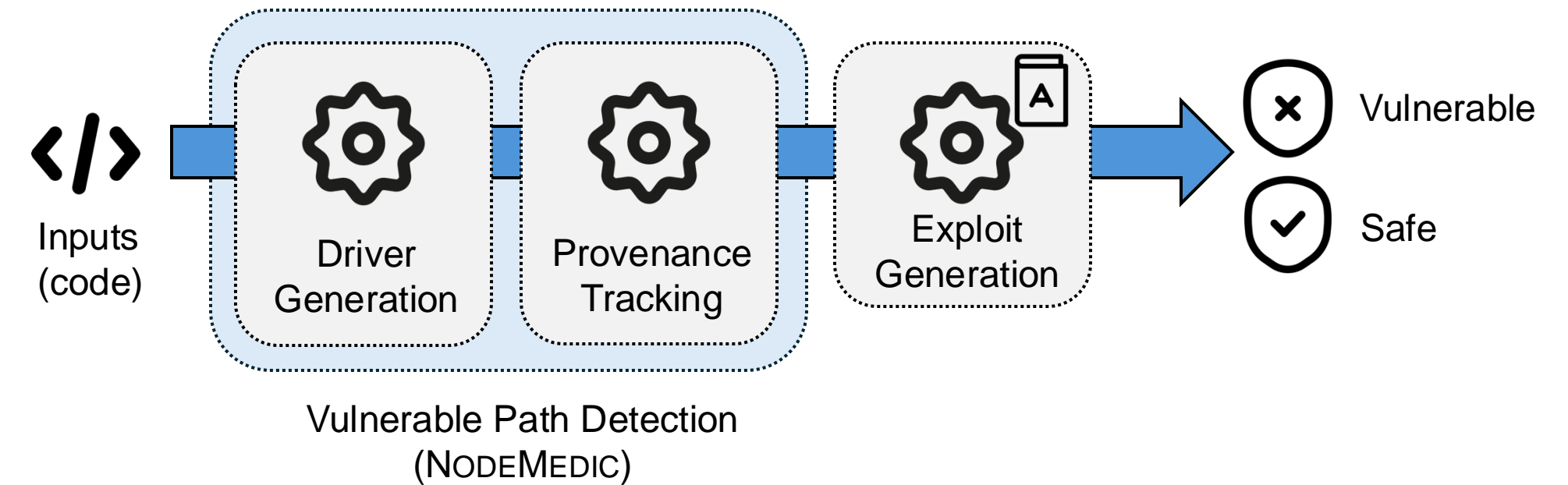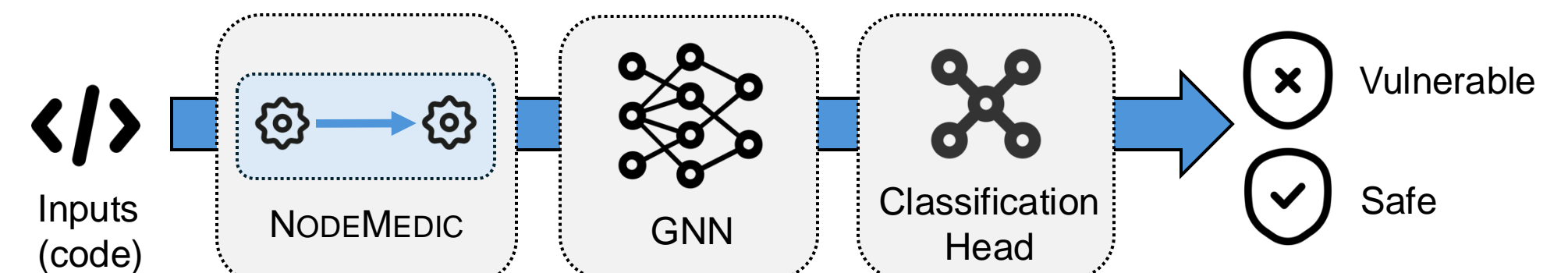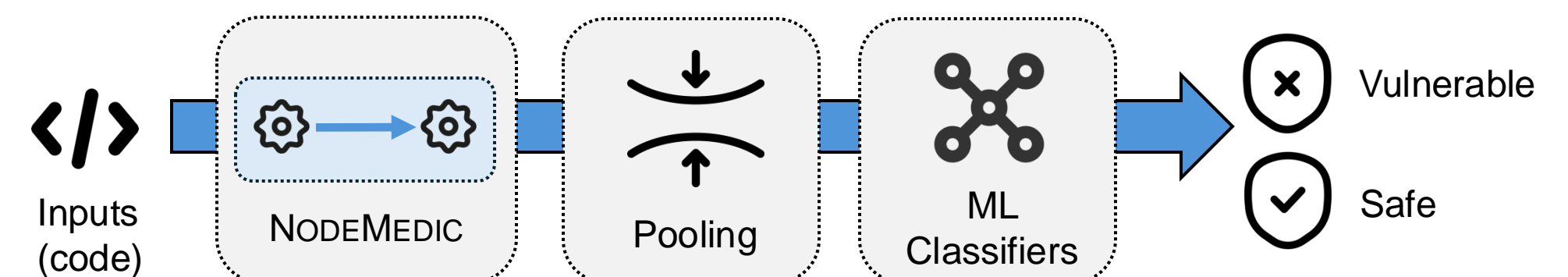
LLM-Program Analysis Hybrid Method

⑨ NODEMEDIC-GNN-LLM

Inputs (code) → NODEMEDIC → GNN → Classification Head → ✕ Vulnerable / ✓ Safe

LLM