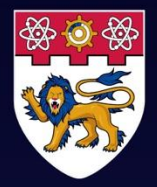


NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Large Language Models for Security Testing

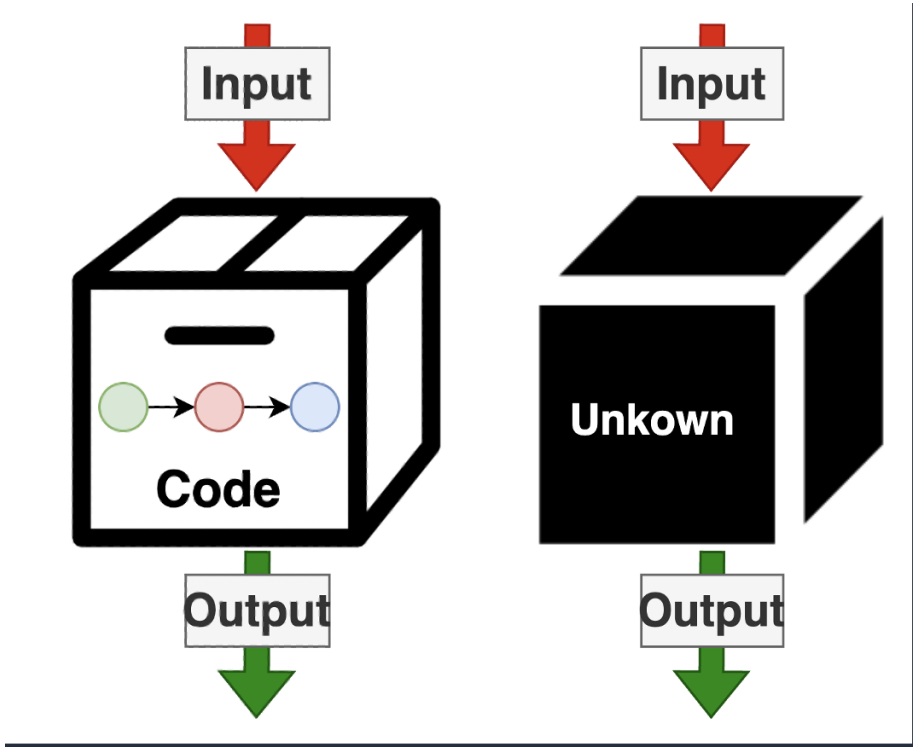
Gelei Deng
Nanyang Technological University, Singapore

The focus of Today's Talk

- Present recent research on LLM-enhanced penetration testing and fuzzing
  - PentestGPT: Evaluating and Harnessing Large Language Models for Automated Penetration Testing
  - HackSynth: LLM Agent and Evaluation Framework for Autonomous Penetration Testing
  - VulnBot: Autonomous Penetration Testing for A Multi-Agent Collaborative Framework
  - PentestAgent: Incorporating LLM Agents to Automated Penetration Testing
  - Fuzz4All: Universal Fuzzing with Large Language Models
- Focus on Framework and Agent Design
- Discuss their advantages and shortcoming, and potential solutions

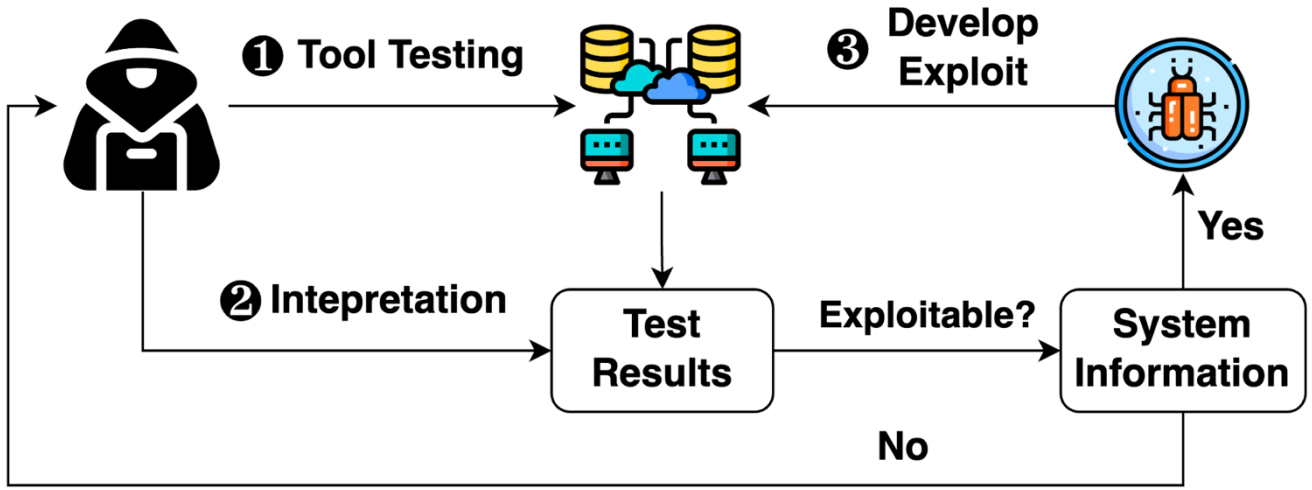
Security Testing – Whitebox vs. Blackbox

- **The process of examining system security with testing**
- **Approaches**
  - **White-box Testing – Audit**
    - Test with knowledge to internals
    - Accurate and efficient
    - **Privacy concerns**
  - **Black-box Testing – Penetration Testing**
    - Test as outsiders
    - Typically requires more domain knowledge
    - **Hard to automate**



Penetration Testing

- **Simulated, controlled cyberattack against a computer system, network, or application to identify vulnerabilities that could be exploited by malicious actors**
- **General Procedure**
  1. Test the system with tools
  2. Interpret testing results
  3. Check Exploitability
     - Develop the exploit, or
     - Go to step 1
- **Key Factor**
  - **The tester's domain knowledge**
    - How to use the tools? How to interpret the test results?
  - **Drawbacks**
    - Not reproducible
    - Rely on penetration tester

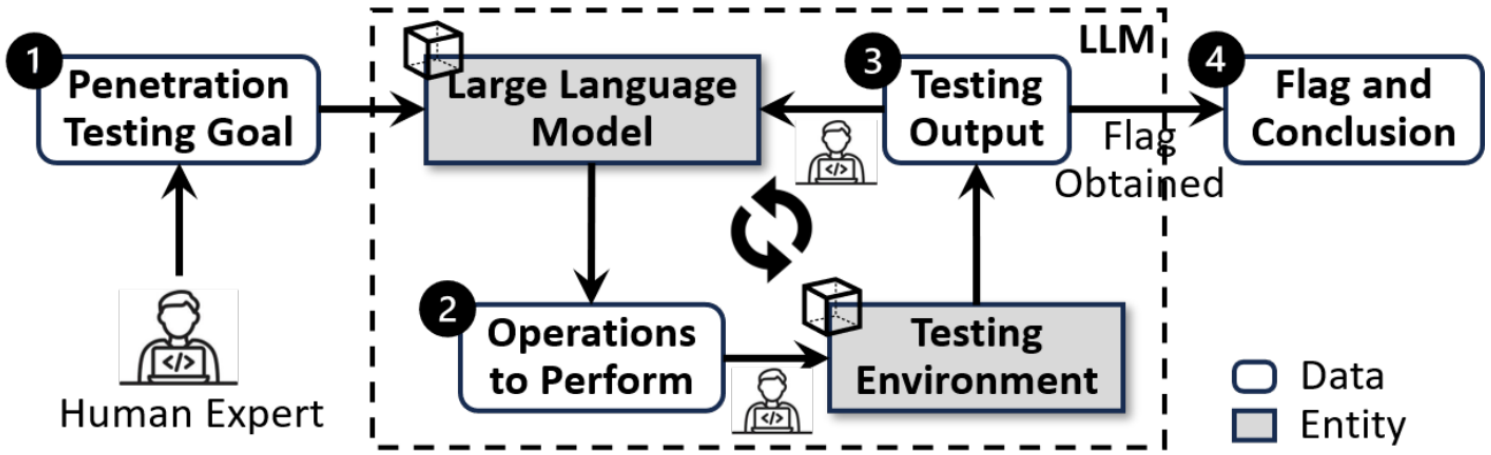


PentestGPT: Evaluating and Harnessing Large Language Models for Automated Penetration Testing

USENIX Security 2024

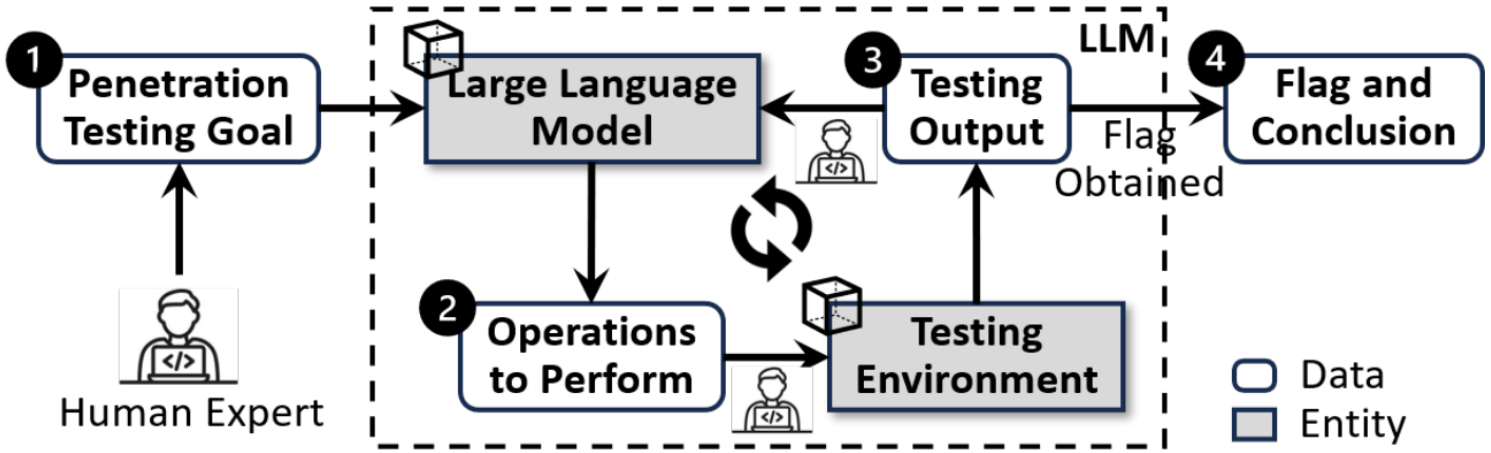
LLM-based Automated Penetration Testing

- **Use LLM as the reasoning module**
- **Human tester without domain knowledge to perform the test**
  - Execute the command generated by LLM
  - Return the outputs back to LLM



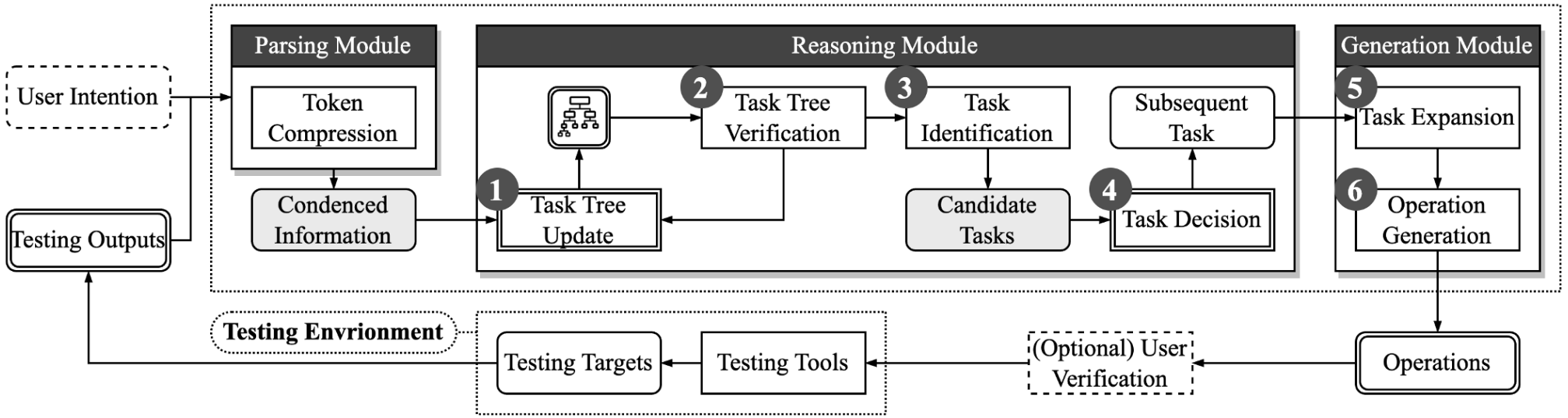
LLM-based Automated Penetration Testing

- **We use a real-world penetration testing benchmark to test different LLMs**
- **Experimental result shows that LLMs suffer from**
  - Limited context window and shifted attention
  - False decision making on long context
  - Hallucination in actual command generation

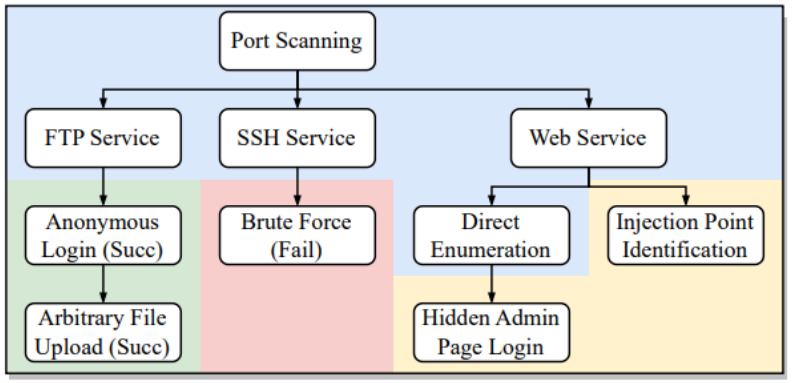


PentestGPT

- **Our LLM-powered penetration testing solution**
  - Optimized to counter the drawbacks of LLMs
    - Token Length Limit – ***Parsing Module***
    - Reasoning Capability – ***Reasoning Module***
    - Hallucination – ***Generation Module***



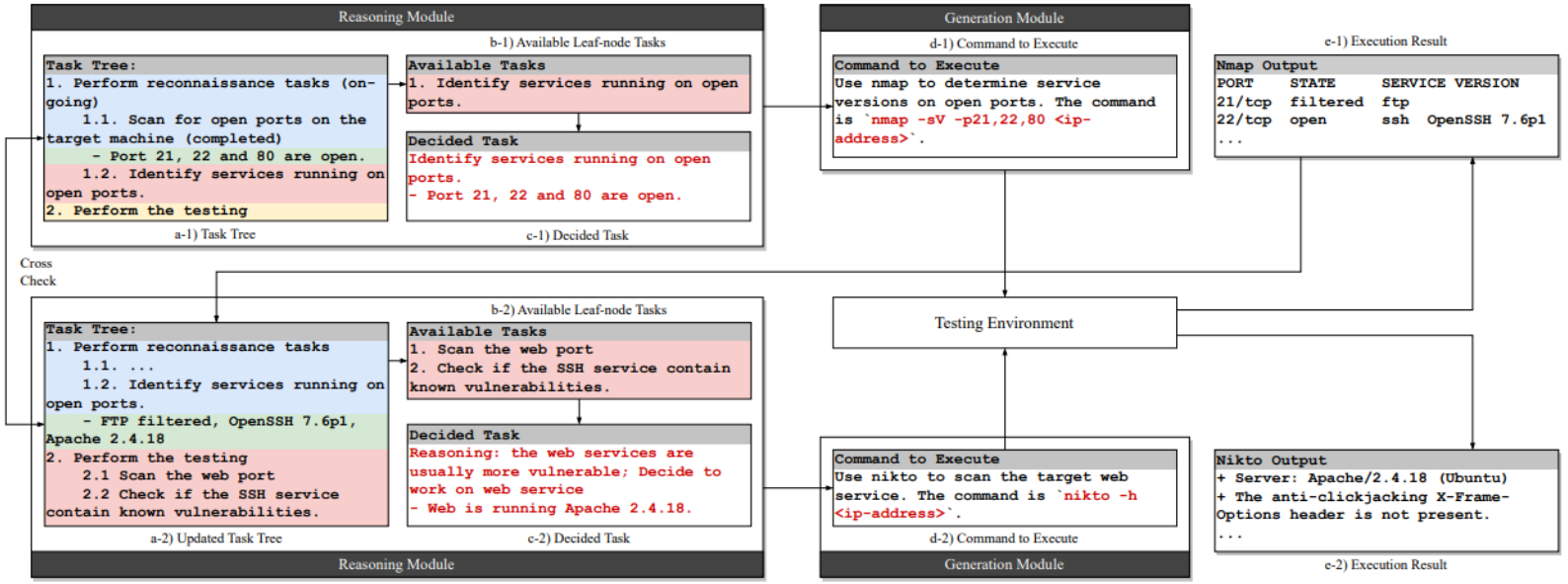
PentestGPT – Penetration Testing Tree Representation



a) PTT Representation

```
Task Tree:
1. Perform port scanning (completed)
  - Port 21, 22 and 80 are open.
  - Services are FTP, SSH, and Web Service.
2. Perform the testing
  2.1 Test FTP Service
    2.1.1 Test Anonymous Login (success)
      2.1.1.1 Test Anonymous Upload (success)
  2.2 Test SSH Service
    2.2.1 Brute-force (failed)
  2.3 Test Web Service (ongoing)
    2.3.1 Directory Enumeration
      2.3.1.1 Find hidden admin (to-do)
    2.3.2 Injection Identification (todo)
```

b) PTT Representation in Natural Language



PentestGPT – Issues and Shortcomings

- **Benchmark Comprehensiveness**
  - Existing benchmarks are not comprehensive in terms of difficulties and categories.
  - Subtasks are not clear enough.
- **Prompt Effectiveness and Transferability**
  - Prompts may not be working on new models (especially from base models to reasoning models)
  - The encoding method of “Penetration Testing Tree” may not be efficient
- **Close-loop Automated Execution**
  - Command execution is not automatic
- **Agentic Framework: tool usage, search and multi-modality capability**
  - Many agent-available features are missing

PentestGPT – Issues and Shortcomings

- **Cumulated Errors in Chain of Process**
  - A minor error in the task will be cascaded into execution failure along with the LLM inference process.
- **Knowledge and Task Encoding**
  - PTT may not be the best encoding method in penetration testing tasks
- **“The bitter lesson”**
  - Should not over-engineer the framework to overcome model shortcomings
  - Instead, assume model will be stronger in reasoning and context handling

HackSynth: LLM Agent and Evaluation Framework for Autonomous Penetration Testing

Lajos Muzsai, David Imolai, András Lukács. https://arxiv.org/abs/2412.01778

Main Contributions

- **HackSynth Agent**
  - Autonomous penetration testing agent powered by Large Language Models
  - A dual-module design: planner and summarizer
- **Evaluation Benchmark**
  - Two CTF-based benchmark sets: PicoCTF and Overthewire platforms
  - 200 challenges in total, covering web exploitation, cryptography and binary exploitation

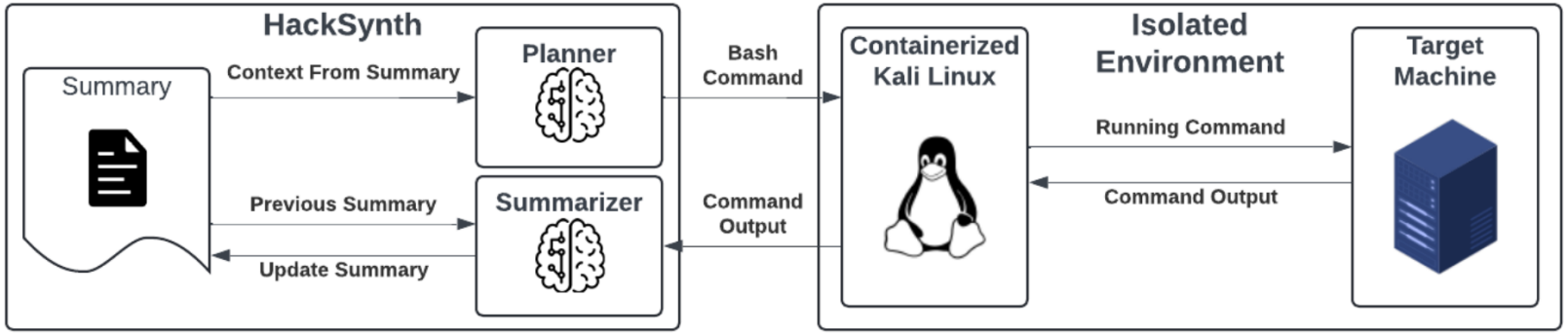


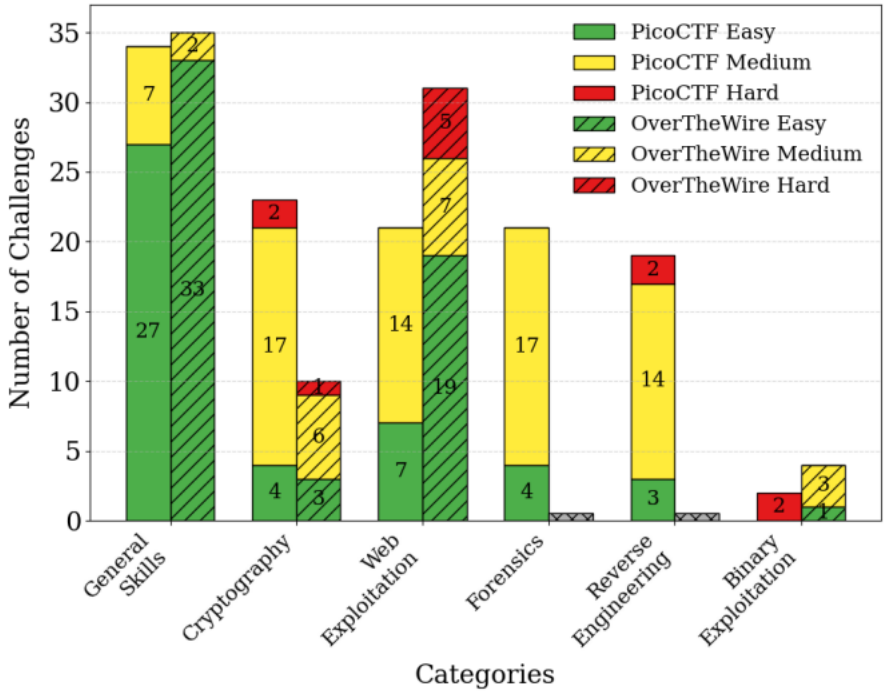
Figure 1: High level overview of the architecture of HackSynth.

HackSynth – Agent Design

- **Dual-module design:**
  - Planner: generates executable commands iteratively
  - Summarizer: processes feedback, tracks hacking state
- **Iterative execution**
  - uses past command context to adapt strategies
  - Tested with multiple LLMs: open-source and proprietary models
  - Best performance with GPT-4o: exceeds system card predictions
  - Safety focus: operates in containerized environment with firewall

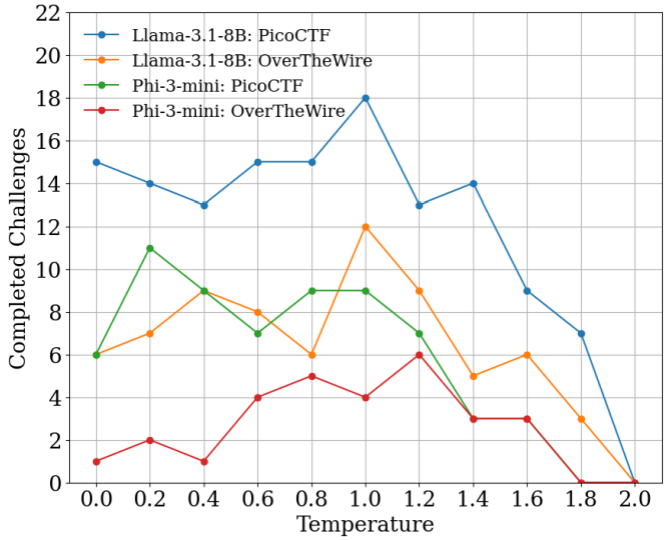
HackSynth – Benchmark Design

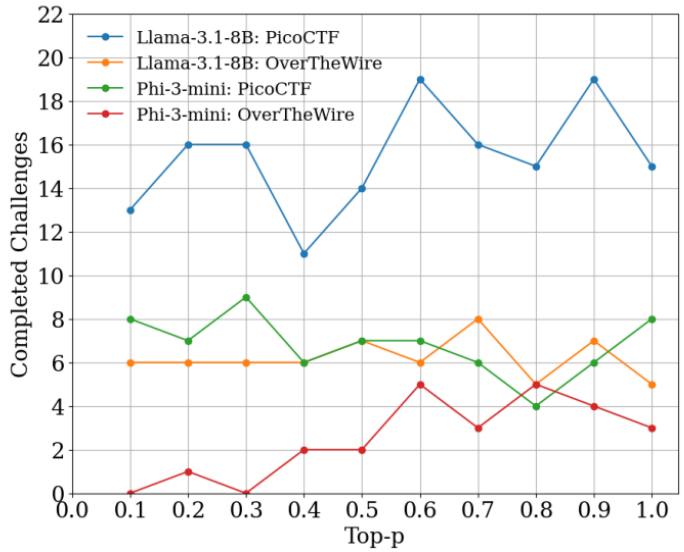
- **Two new CTF-based benchmark sets: PicoCTF and OverTheWire platforms**
  - 200 challenges total: covers web exploitation, cryptography, binary exploitation, etc. *The number is significantly more than existing bencmarks*
  - Difficulty levels: easy, medium, hard

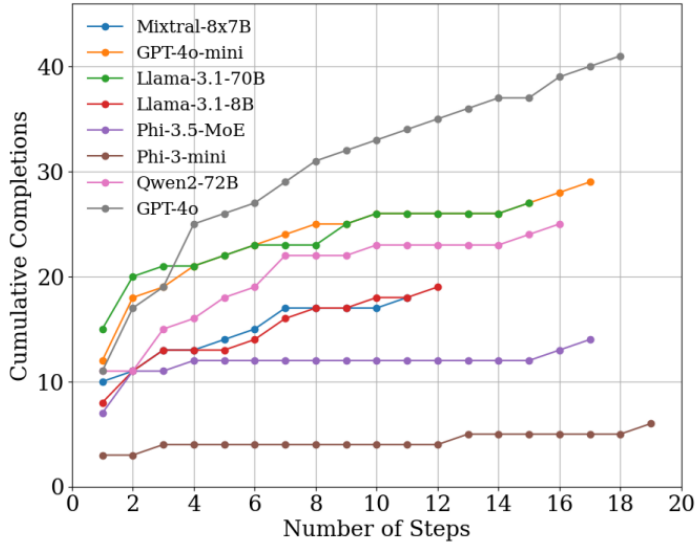


HackSynth – Benchmark Design

- **Two new CTF-based benchmark sets: PicoCTF and OverTheWire platforms**
- **Experiments analyze:**
  - Creativity parameters: temperature and top-p settings
  - Token utilization: efficiency in command generation
  - Model comparison: performance of models on different tasks





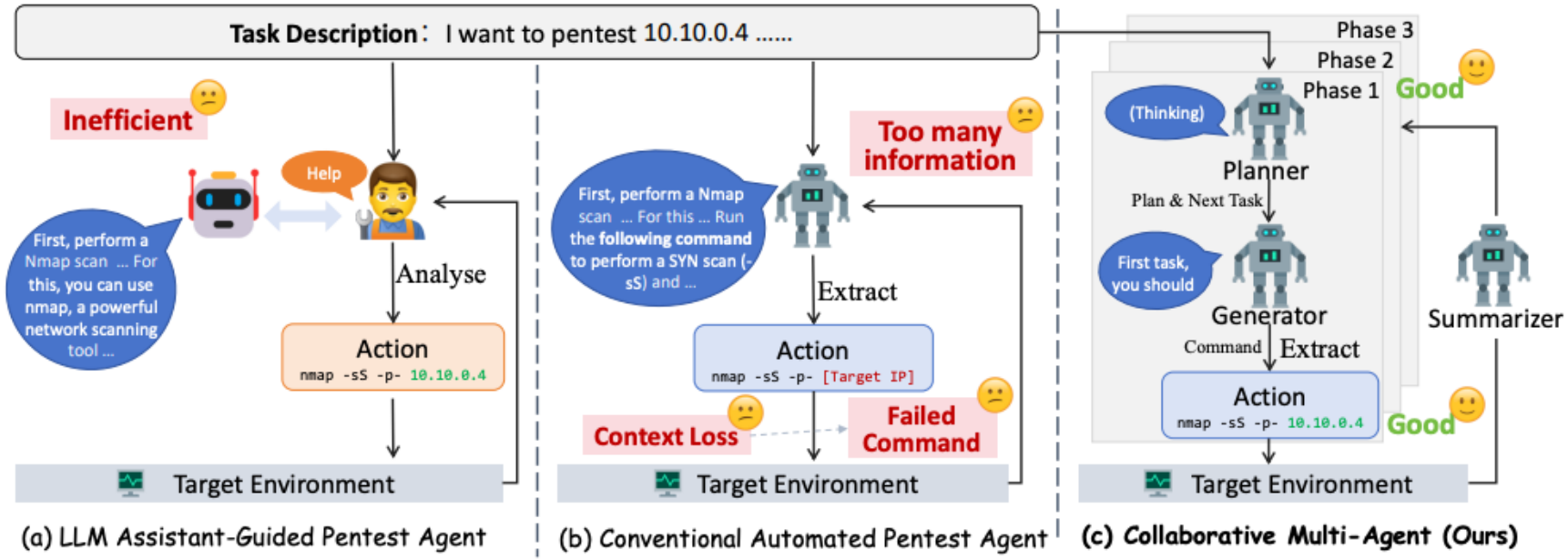


VulnBot: Autonomous Penetration Testing for a Multi-Agent Collaborative Framework

He Kong, Die Hu, Jingguo Ge, Liangxiong Li, Tong Li, Bingzhen Wu. https://arxiv.org/abs/2501.13411

VulnBot – Agent Design

- **A collaborative Agent Framework**
  - Planner: plan for the next task
  - Generator: generate the exact command
  - Summarizer: summarize feedback from target environment.

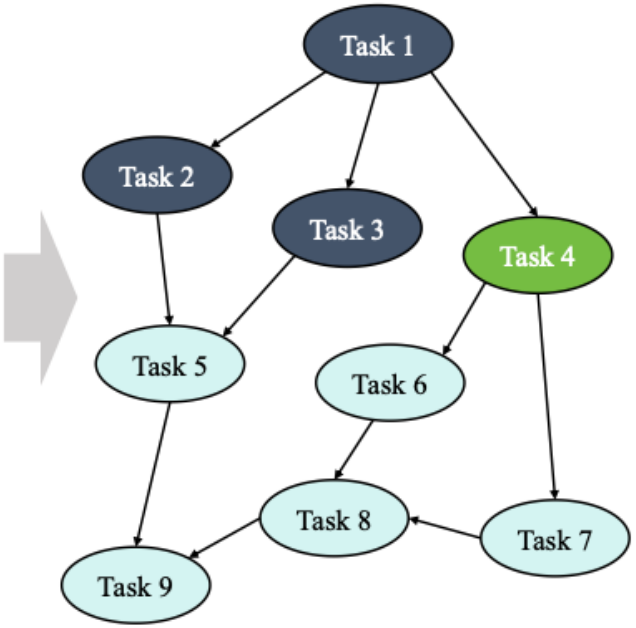


VulnBot – Agent Design

- **Using Penetration Task Graph for task representation**
  - PTG is designed to be task driven
  - Each task is depended on others
  - Directly encode operations
    - Shell execution
    - GUI operation
    - Manual

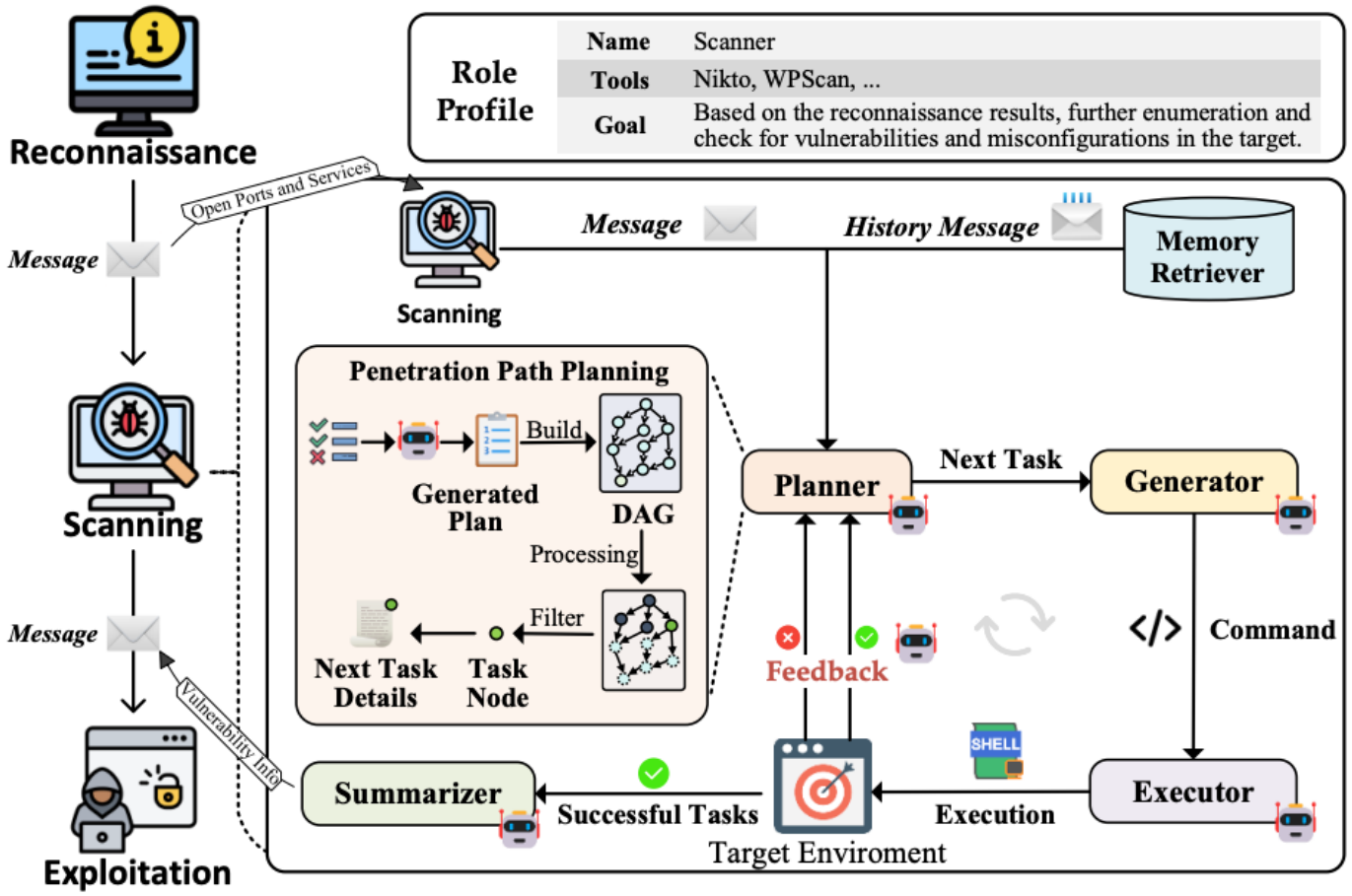
Task List


```
{
  "id": "1", "dependencies": [],
  "instruction": "Use the credentials (wavex:door+open) to SSH into the target machine (IP: 192.168.1.104, Port: 22.",
  "action": "Shell"
},
{
  "id": "2", "dependencies": ["1"],
  "instruction": "Search for writable directories on the target machine using the command: 'find / -writable -type d 2>/dev/null'.",
  "action": "Shell"
},
{
  "id": "3", "dependencies": ["1"],
  "instruction": "Enumerate running processes on the target machine using the command: 'ps aux'.",
  "action": "Shell"
},
......
{
  "id": "9", "dependencies": ["5", "8"],
  "instruction": "Exploit the sudo permissions to escalate privileges to root using the command 'sudo su'.",
  "action": "Shell"
}
```

VulnBot – Agent Design

- **General Design**
  - Inter-agent communication is through natural language
  - Use summarizer to bridge different agents
  - Memory Retrieval through RAG

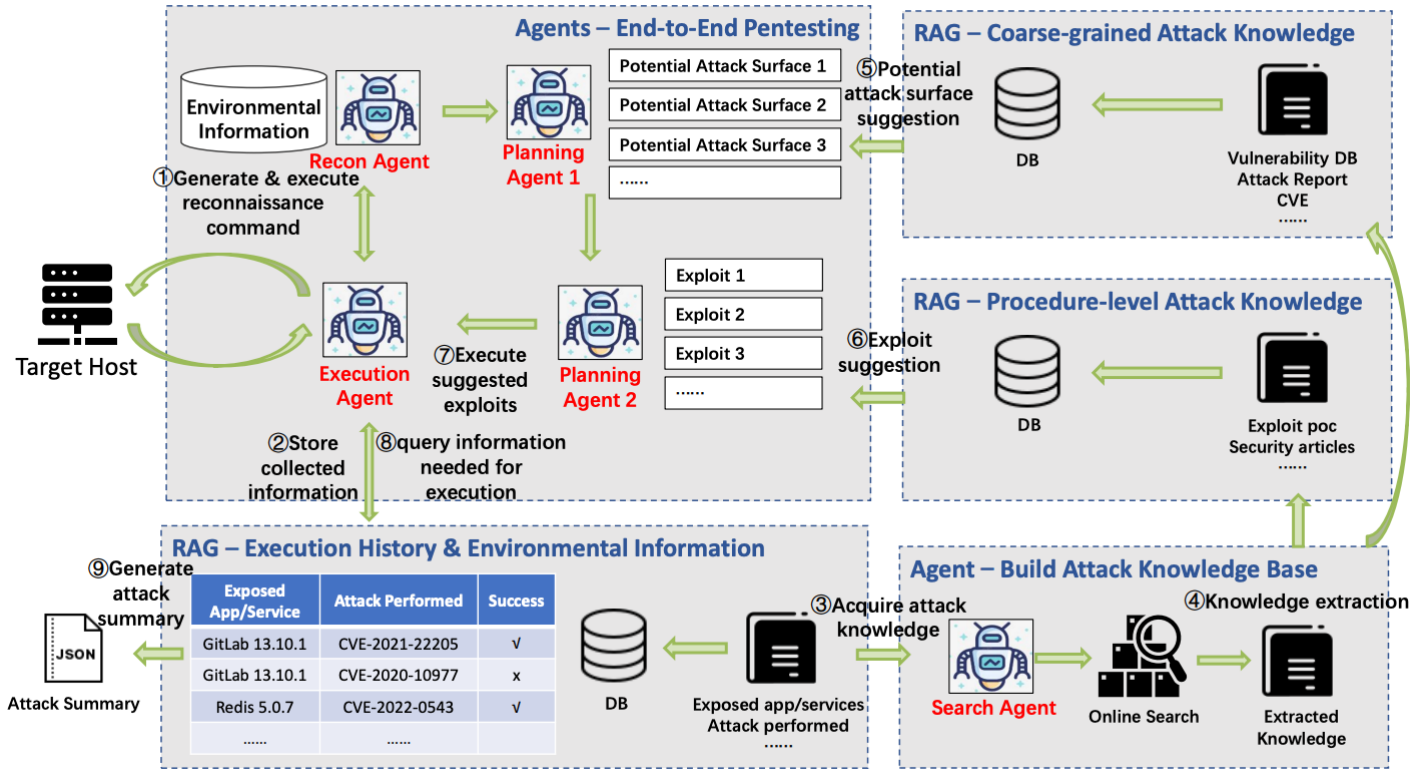


PentestAgent: Incorporating LLM Agents to Automated Penetration Testing

Xiangmin Shen, Lingzhi Wang, Zhenyuan Li, Yan Chen, Wencheng Zhao, Dawei Sun, Jiashui Wang, Wei Ruan,
https://arxiv.org/pdf/2411.05185v1

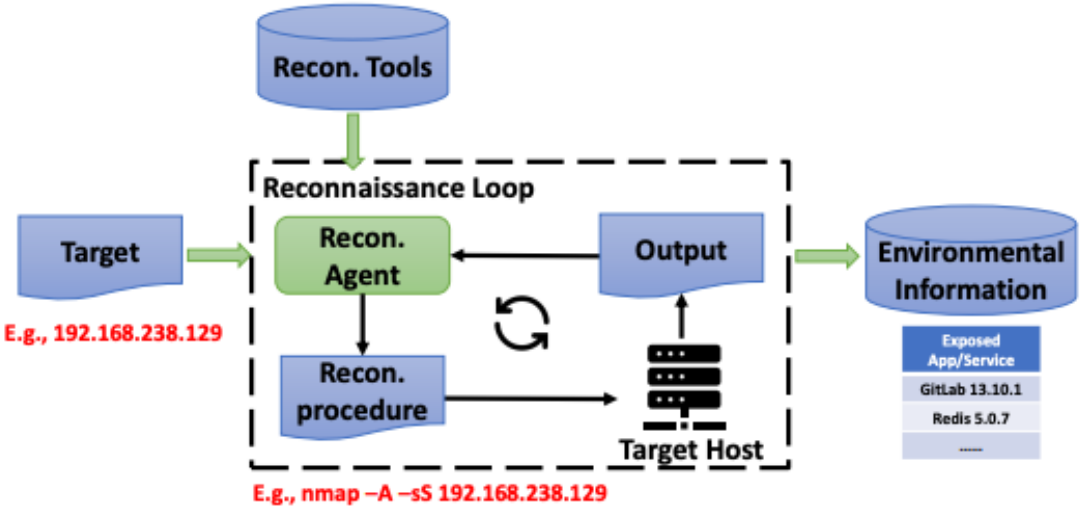
PentestAgent – Agent Design

- **General Design**
  - Three types of agents
    - Recon
    - Planning
    - Execution
  - Use RAG for data retrieval
  - Build Attack Knowledge Base



PentestAgent – Message Design

- **Design prompts with *Role, CoT, Context Knowledge, Structured Output***
  - Transferrable to all agents
  - Provide context-related knowledge for different tasks
    - RAG
    - Few-shot examples



Reconnaissance System Message (Simplified)

Role-play

You're an excellent cybersecurity penetration tester assistant. Guide the tester …

Chain-of-Thought

Use Nmap to identify exposed ports, then use relevant tools in Nmap to analyze these ports on the target host …

RAG

You should use your query tool to learn about available reconnaissance tools …

Structured Output

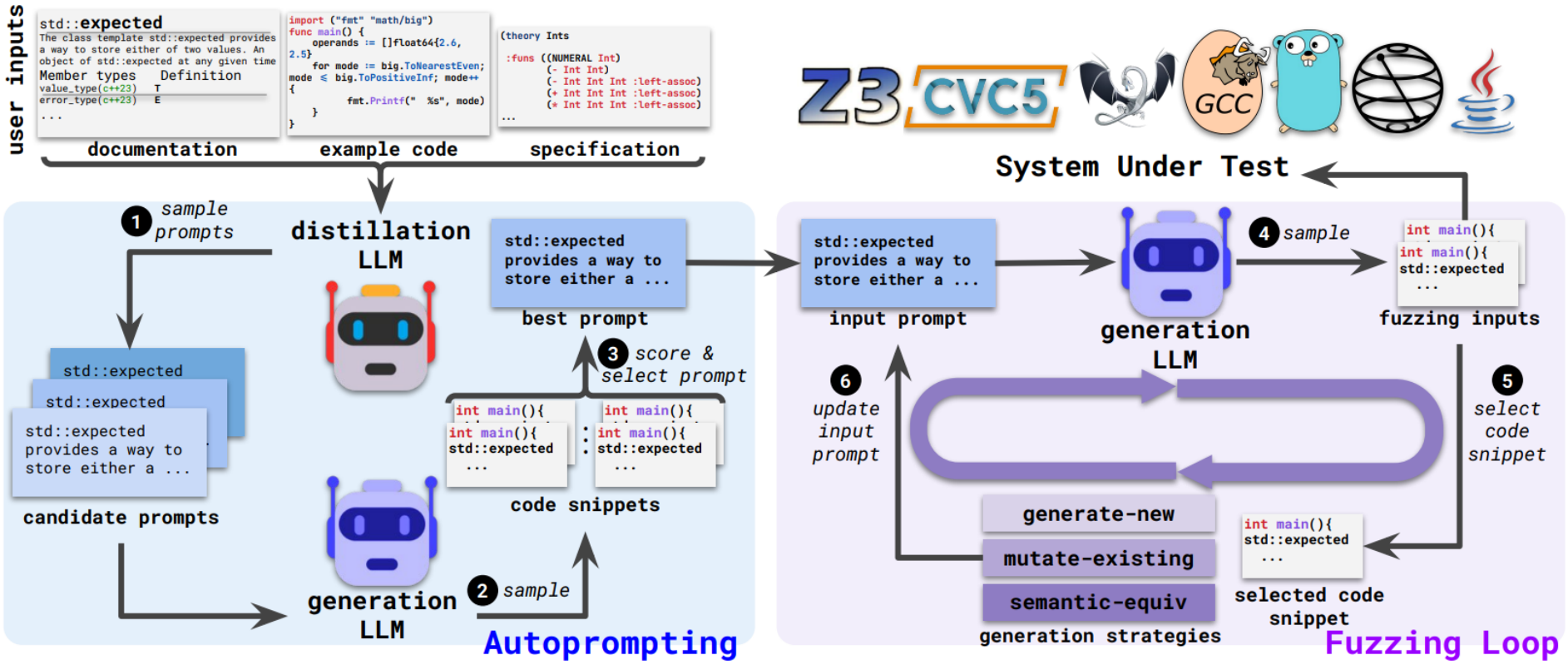
You should always respond in valid JSON format with the following fields: {FORMAT SPEC.} …

Fuzz4All: Universal Fuzzing with Large Language Models

Chunqiu Steven Xia, Matteo Paltenghi, Jia Le Tian, Michael Pradel, Lingming Zhang. https://arxiv.org/abs/2308.04748

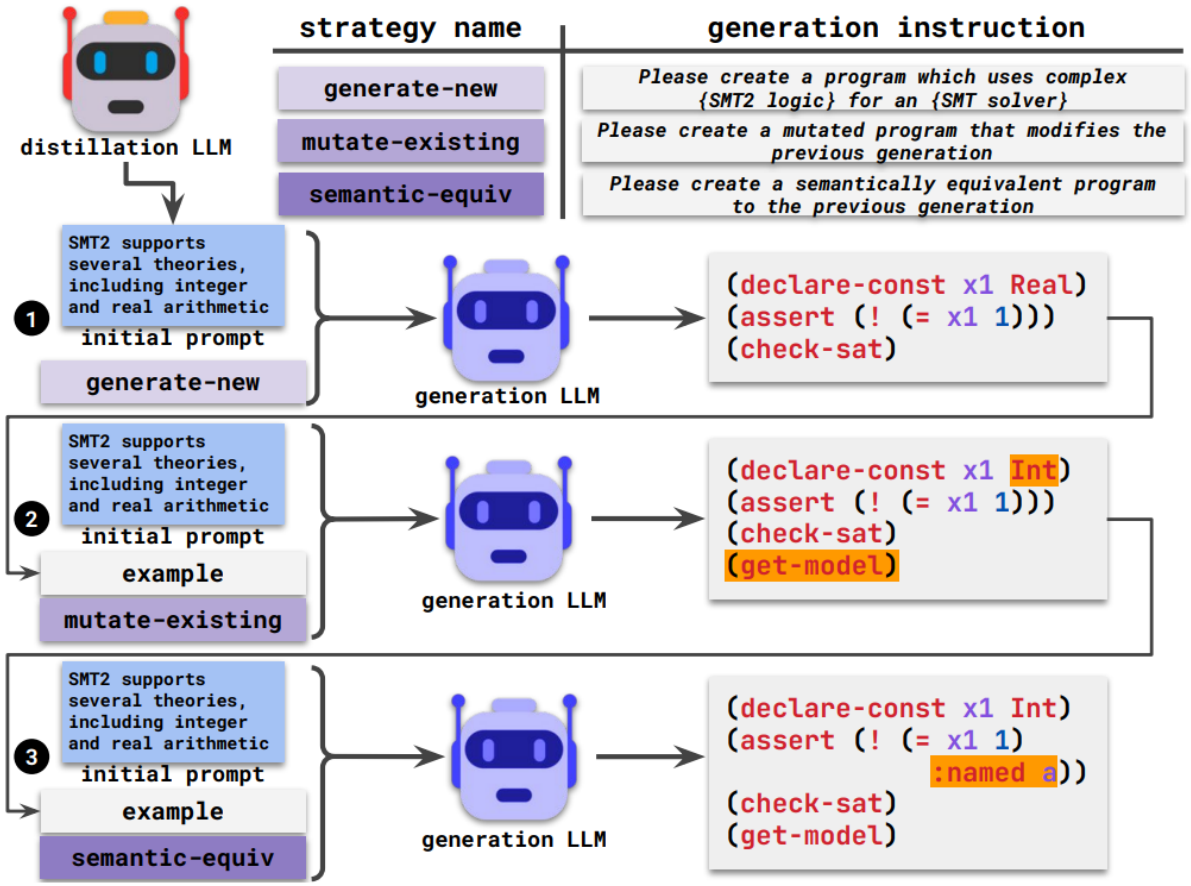
Fuzz4All – General Design

- The first universal fuzzer targeting different languages.
- Using LLMs for input generation and mutation.



Fuzz4All – General Design

- **Design fuzzing strategies into LLM prompts.**
- **Decouple strategies with the actual generation process**
  - Reduces hallucination
  - Can be easily expanded with heuristics
  - Reduce cascaded errors

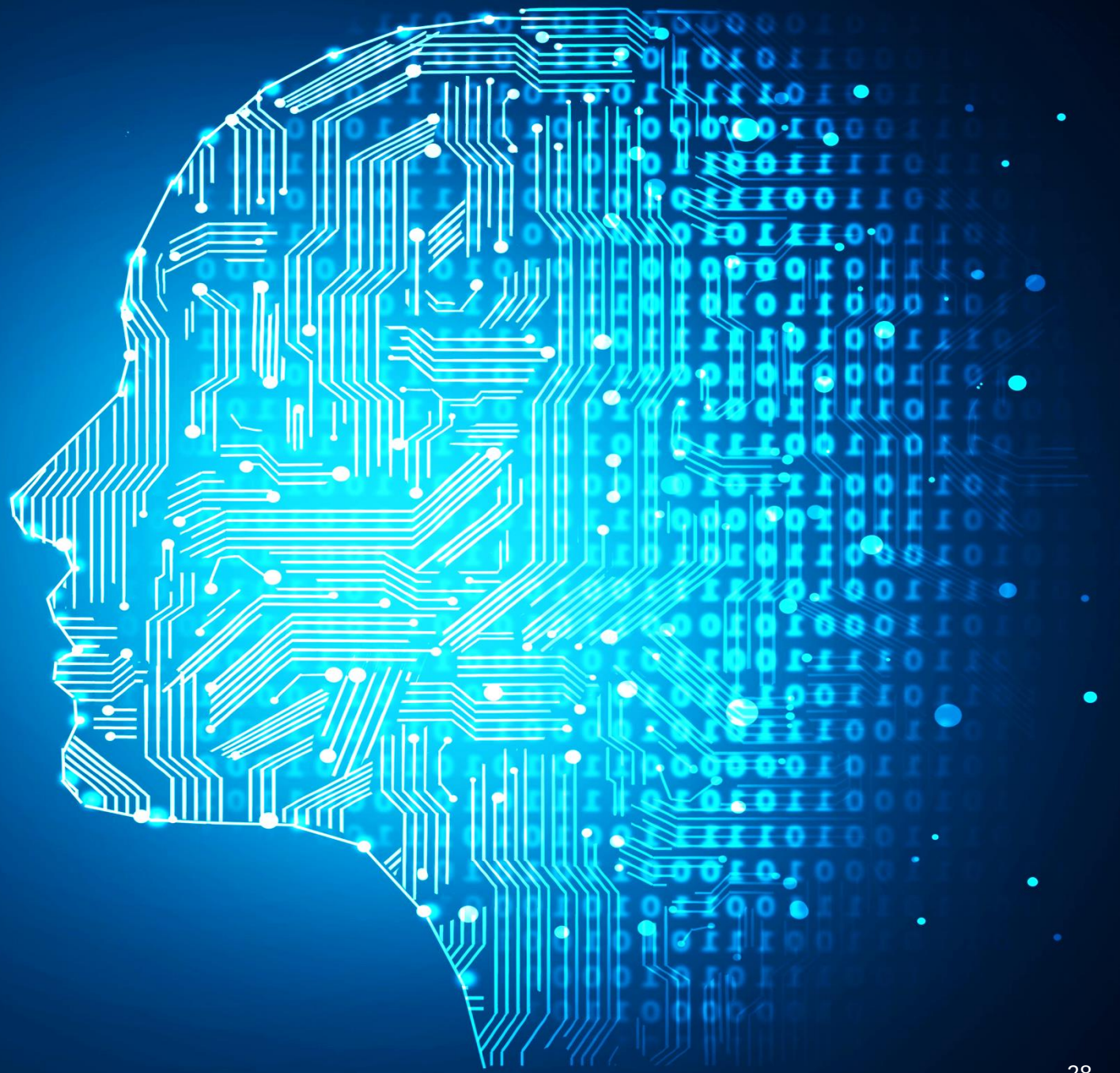


Revisiting Drawbacks of PentestGPT Design

- **Benchmark Comprehensiveness** (HackSynth)
- **Prompt Effectiveness and Transferability**
- **Close-loop Automated Execution** (Vulnbot, PentestAgent)
- **Agentic Framework: tool usage, online search and multi-modality capability**
- **Cumulated Errors in Chain of Process** (Fuzz4all proposes a stratregy)
- **Knowledge Encoding**

THANK YOU!

Q&A



NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE