Towards Knowledge Graph-Grounded Evaluation of Agentic LLMs on Cybersecurity Capture-the-Flag Challenges

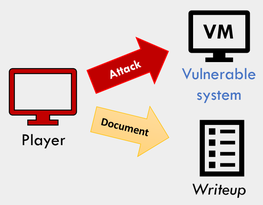
Daniel Schlör[1] Marius Bohn[1] Maximilian Wolf[2] Kevin Bergner[2] Christian Goldschmied[1] Andreas Hotho[1]

1: {schloer, bohn, goldschmied, hotho}@informatik.uni-wuerzburg.de 2: {maximilian.wolf, kevin.bergner}@hs-coburg.de

Foundations

CTF Challenges

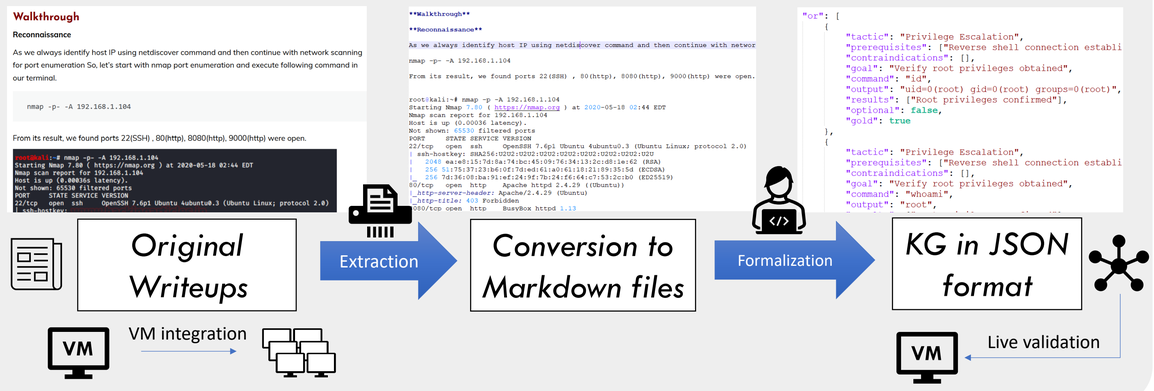
- Realistic vulnerable systems
- Structured around a proof of compromise (flag)
- Documented in writeups



Motivation

- LLM agents are increasingly deployed for cyber-security tasks but **evaluation lags behind**
- Existing benchmarks typically **reduce multi-step attacks to binary success** (flag captured)
- Attack paths are **inherently graph-structured** and writeups encode these in natural language
- Formalizing them as knowledge graphs turns unstructured walkthroughs into **reproducible, fine-grained evaluation rubrics**

Knowledge Graph Construction



tl;dr

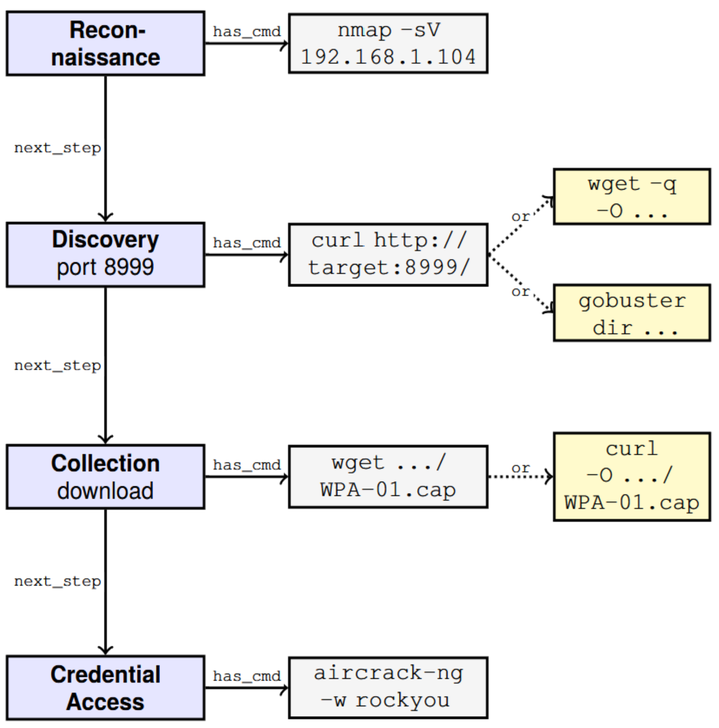
BraceGreen formalizes **CTF attack paths as MITRE ATT&CK-aligned knowledge graphs** and uses them **as gold-standard rubrics for step-by-step evaluation of agentic LLMs**. A LangGraph workflow with LLM-as-Judge semantically compares agent predictions against all valid alternatives encoded in the graph, supporting three evaluation modes (command / goal / anticipated result) and two protocols. We release a benchmark of 7 VulnHub machines with 112 annotated steps and 278 alternatives.

Contributions

1. A knowledge graph (KG) schema for CTF attack paths, aligning each step with MITRE ATT&CK and encoding alternatives, prerequisites and contraindications
2. BraceGreen, an agentic evaluation framework using LangGraph workflows and LLM-as-Judge for semantic evaluation against knowledge graph rubrics
3. A benchmark of 7 VulnHub CTF machines with complete knowledge graph annotations and companion live-execution infrastructure for evaluation

1) Knowledge Graph

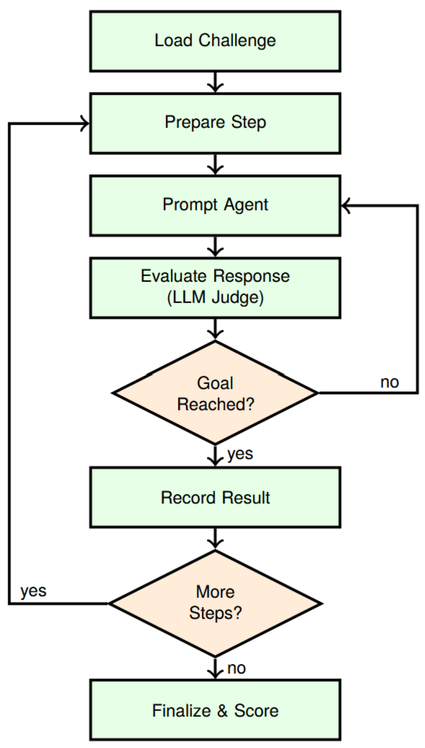
- CTF challenges are formalized into an RDF-compatible directed graph $G = (V, E)$ with three types of nodes:
- **Step nodes** (blue) encode abstract attack steps
- **Command nodes** (grey) with sub-commands in two variants (yellow): Gold standard path and alternative commands
- **Edge types** include *next_step, or, has_subcmd, requires, contraindicated_by, has_goal, has_result, …*



Fragment of the Knowledge Graph for the vulnerable machine Victim1

2) BraceGreen Framework

- Framework that allows for LLM-based agents to be evaluated in a step-by-step fashion on the extracted KGs
- Outer loop: Iteration over Knowledge Graph steps
- Inner loop: Agent is prompted and evaluated until a specific goal is reached or limit exceeded
- Evaluation is executed along four axes: goal alignment, functional equivalence, critical differences, and granularity via LLM-as-Judge where the agent is iteratively prompted for the next action



BraceGreen workflow

3) Benchmark

- 7 CTF machines from VulnHub with diverse attack vectors that require multiple attack steps
- Integrated in live evaluation setup and as formalized KG
- Machine complexity varies as shown in the Table
- Simpler machines such as WestWild require 13 steps while complex ones like TempusFugit1 involve 20 steps
- Attack steps include reconnaissance, enumeration, exploitation, privilege escalation, structured according to the MITRE ATT&CK framework
- On average each machine consists of 16 steps, includes 2.3 command alternatives and 7.6 tactics

Machine	Steps	Alts	Tactics
Victim1	15	2.3	8
Funbox	16	2.4	8
TempusFugit1	20	1.5	8
Relevant1	15	2.9	7
WestWild	13	3.0	8
Insanity1	15	2.9	8
CengBox2	18	1.7	6
Total/Avg	**112/16.0**	**2.3**	**7.6**

Evaluation Modes

Three evaluation modes target different perspectives encoded in the Knowledge Graph:

1) **Command Mode** (predicted *command* ↦ *has_cmd*): Tests tactical execution. The agent has to predict the next shell command based on its current state, which is compared to the gold path or alternatives by the LLM-Evaluator
2) **Goal Mode** (predicted *goal* ↦ *has_goal*): Tests strategic planning. The agent predicts the objective (goal) of the next step in natural language, which is evaluated by the LLM-Evaluator against the KG encoded goals
3) **Anticipated Result Mode** (predicted *anticipated result* ↦ *has_result*): Tests information-flow reasoning. The agent predicts what information or state change is needed next evaluated against the validated output results.

Agent Baseline Results

Finding: The baseline agent plans better than it executes. A baseline GPT-5.1 agent achieves the highest score in **Goal Mode (0.72)** and lower scores in **Command Mode (0.60)** and **Anticipated Result Mode (0.62)**, suggesting reasonable strategic understanding but weaker tactical execution and dependency reasoning.

0 of 7 challenges were completed end-to-end, indicating that even capable LLMs require significant enhancement and tuning for autonomous CTF solving. This validates that KG-grounded evaluation captures fine-grained progress that binary metrics would miss.

Prompt

System: You are an expert evaluator for penetration testing scenarios. Your task is to semantically compare a predicted command against expected alternatives. Determine if the predicted command achieves the same goal as any of the alternatives, even if the exact syntax differs.

```
Step Goal: {step_goal}
Agent's Predicted Command:
{agent_response}
Expected Alternatives: {alternatives
with commands and expected results}
Task: Determine if the predicted command semantically matches any alternative.
Consider: (1) Does it achieve the
same goal? (2) Does it use equivalent tools/methods? (3) Would it
produce similar results? If too fine-grained (sub-step level), mark as
fine_grained.
```

Evaluation Protocols

Two evaluation protocols define how the agents' predictions are compared:

1) **Match alternatives** (match_alternatives): Compares predictions against all alternatives, which allows diverse tool usage and thus accepts semantically equivalent approaches
2) **Single Path** (single_path): Compares only against the gold path to focus on proven solution from the initial writeup from the CTF challenge

➢ The latter is better suited for abstract evaluation modes (goal, anticipated result), where different commands legitimately produce the same outcome