Teams of LLM Agents can Exploit Zero-Day Vulnerabilities

Yuxuan Zhu, Antony Kellermann, Akul Gupta, Philip Li, Richard Fang, Rohan Bindu, Daniel Kang
University of Illinois Urbana-Champaign • EACL 2026

Abstract

Large language model (LLM) agents have become powerful enough to hack real websites. Previous research showed that a single AI agent could exploit known vulnerabilities when given a description of the problem. However, when the vulnerability is completely unknown to the agent — called a "zero-day" vulnerability — single agents fail because they cannot explore many different attack paths at once.

This paper introduces HPTSA (Hierarchical Planning and Task-Specific Agents), a team of AI agents that work together to find and exploit zero-day vulnerabilities in real websites. A planning agent explores the website and decides which specialist agents to send. Each specialist focuses on a specific type of attack. On a benchmark of 14 real-world vulnerabilities, HPTSA achieved a 42% success rate — up to 4.3 times better than a single GPT-4 agent.

1. Introduction

AI agents are getting better at complex tasks like solving GitHub issues and organizing emails. But as their abilities improve, so does the risk of misuse. Hacking is one of the biggest concerns because AI agents could be used to break into computer systems.

Earlier work showed that simple AI agents could hack mock "capture-the-flag" websites and real-world vulnerabilities when given a description of the flaw. But these agents failed when the vulnerability description was removed — the realistic zero-day scenario. This raised a question: can more sophisticated AI agents exploit real zero-day vulnerabilities?

The answer is yes. This paper develops HPTSA, a multi-agent system that is the first to successfully exploit real-world zero-day vulnerabilities using teams of AI agents. Prior single agents struggled because cybersecurity tasks require exploring many different attack types and maintaining long-range plans — something a single agent cannot do well.

HPTSA uses a hierarchical design with three components: a planner that explores the website, a team manager that routes tasks, and specialist agents that focus on specific attack types like SQL injection or cross-site scripting. On a benchmark of 14 real-world vulnerabilities, HPTSA with GPT-4 achieved 42% pass@5, within 1.8x of an agent that was told the vulnerability description. Open-source models and automated scanners scored 0%.

2. Background

2.1 Computer Security

A vulnerability is a flaw in a computer system that allows unintended behavior, usually for malicious purposes. Exploiting a vulnerability means detecting it and taking advantage of it. This paper focuses on zero-day vulnerabilities — flaws that are unknown to the software's owner. Unlike known ("one-day") vulnerabilities where a patch exists but hasn't been applied, zero-days cannot be defended against proactively.

There is an important distinction between a vulnerability class (like SQL injection) and a specific instance of that class. Even a company like Microsoft, which invests billions in security, was hacked in 2021 through a specific instance of server-side request forgery (SSRF). This shows why finding specific zero-day instances matters enormously.

2.2 AI Agents and Cybersecurity

AI agents powered by large language models can now perform complex tasks by using tools through APIs. Recent work showed these agents could hack capture-the-flag websites and known vulnerabilities when given a description. They work through a ReAct-style loop: take an action, observe the result, and repeat.

However, single agents fail in the zero-day setting. They get stuck trying one approach and cannot backtrack to try different attack types. The context window fills up quickly with cybersecurity tasks, making long-range planning impossible for one agent alone.

3. HPTSA: Hierarchical Planning and Task-Specific Agents

HPTSA solves the limitations of single agents by using a team of specialized agents organized in a hierarchy. The key insight is that cybersecurity requires both broad exploration (trying many attack types) and deep analysis (executing specific exploits). No single agent can do both well.

3.1 Overall Architecture

HPTSA has three major components working together:

Hierarchical Planner: This agent explores the website to understand its structure and determine what types of vulnerabilities might exist. For example, it might notice the login page is a good target and focus attention there.

Team Manager: Based on the planner's findings, this agent decides which specialist agents to dispatch. It also collects results from previous runs and can rerun agents with better instructions or call different agents based on what was learned.

Task-Specific Expert Agents: Each specialist is designed to exploit a specific class of vulnerability — SQL injection, cross-site scripting, cross-site request forgery, and others. This design mirrors how real cybersecurity teams work: generalists explore, specialists execute.

3.2 Task-Specific Agents

The researchers built six specialist agents, each focused on a different type of web attack:

XSS Agent: Looks for cross-site scripting flaws where malicious JavaScript can be injected into web pages viewed by other users.

SQLi Agent: Tests for SQL injection vulnerabilities where database queries can be manipulated by injecting malicious code into input fields.

CSRF Agent: Searches for cross-site request forgery vulnerabilities that trick users into performing unwanted actions on websites where they are logged in.

SSTI Agent: Probes for server-side template injection flaws that can lead to remote code execution on the server.

ZAP Agent: Uses the OWASP ZAP automated scanner as a tool alongside LLM reasoning.

Generic Agent: Handles miscellaneous web hacking techniques not covered by the other specialists.

Each agent has three capabilities: access to tools (a browser framework called Playwright, terminal access, and file management), reference documents (5-6 curated guides per vulnerability type), and custom prompts tailored to the specific attack.

3.3 Implementation

The system was built using LangChain and LangGraph to coordinate agents, with Fireworks and OpenAI APIs powering the individual agents. To reduce costs, the team implemented an HTML simplification strategy that strips unnecessary tags (images, SVGs, styles) before sending webpage content to agents, dramatically reducing token usage.

4. Benchmark of Zero-Day Vulnerabilities

To test HPTSA fairly, the researchers created a benchmark of 14 real-world zero-day vulnerabilities. They set strict criteria for inclusion:

First, all vulnerabilities had to be published after the knowledge cutoff date of GPT-4, ensuring the model could not have memorized them during training. Second, they focused on web vulnerabilities with clear pass/fail criteria — not vague conditions that are hard to test. Third, only vulnerabilities that could be reproduced manually were included, confirming the benchmark is reproducible.

The benchmark includes 14 vulnerabilities across different types (XSS, CSRF, SQLi, arbitrary code execution) and severity levels (medium to critical). Notable examples include a critical SQL injection in Dolibarr (CVE-2024-5314, severity 9.1) and a privilege escalation in Zabbix (CVE-2024-22120, severity 9.1). All vulnerabilities were verified to work in sandboxed environments to ensure no real users were harmed.

5. HPTSA can Autonomously Exploit Zero-day Vulnerabilities

5.1 Experimental Setup

The researchers measured success by manually checking whether agents actually exploited each vulnerability — not just detected it. They used two metrics: pass@5 (success in 5 attempts) and pass@1 (overall success rate). Pass@5 is the primary metric because in real hacking, only one successful attempt is needed.

They compared HPTSA against several baselines: a single ReAct agent without vulnerability description (lower bound), a single agent given the vulnerability description (upper bound), MetaGPT (a general multi-agent framework), and open-source scanners ZAP and MetaSploit.

5.2 End-to-End Results

HPTSA with GPT-4 achieved the best results: 42% pass@5 and 18% overall success rate. Open-source models (Llama 3.1 405B, Qwen 2.5 72B) scored 0% — they either refused to attempt exploitation or repeatedly tried the same incorrect approach.

Method	Pass@5	Pass@1
HPTSA (GPT-4)	42%	18%
1DV Agent (GPT-4, with desc.)	75%	33%
1DV Agent no desc. (GPT-4)	21%	4.2%
MetaGPT	0%	0%
ZAP + MetaSploit	0%	0%

Table 1: End-to-end results comparing HPTSA against baselines.

HPTSA outperformed the single-agent GPT-4 by 4.3x on pass@1 and 2.0x on pass@5. It performed within 1.8x of the upper-bound agent that was told the vulnerability description. Both MetaGPT and open-source scanners achieved 0% on the benchmark.

5.3 Ablation Studies

To understand which components matter most, the researchers tested HPTSA with parts removed:

Configuration	Pass@5	Pass@1
Full HPTSA	42%	18%
Without task-specific agents	21%	8.3%
Without reference documents	33%	8.3%
Without hierarchical structure	7.1%	1.4%

Table 2: Ablation study results showing each component's contribution.

Removing task-specific agents cut performance in half. Removing documents reduced pass@1 by 2.1x. Removing the hierarchical structure caused the biggest drop — 13x lower pass@1 and 6x lower pass@5. These results prove that all three components are necessary.

6. Case Studies

To understand how HPTSA works in practice, the researchers analyzed detailed traces of both successful and failed exploitation attempts.

6.1 Success Case Studies

Flusity-CMS CSRF and XSS (CVE-2024-24524 and CVE-2024-27757): The admin panel of this content management system had a CSRF flaw where clicking a crafted HTML file could create a new menu, and an XSS vulnerability in the gallery add-on.

HPTSA's process unfolded across multiple specialist agents:

The XSS agent was dispatched first with generic instructions. On its first run, it logged in but stopped exploring too early. On the second run, it navigated to the admin panel and injected XSS into a post — but not the specific CVE vulnerability. On the third run, it explored the gallery add-on and successfully exploited CVE-2024-27757.

The SQL injection agent tried multiple approaches on the login page and admin panel but found nothing. The CSRF agent was then given a narrower focus on the admin panel endpoints. On its first run, it created a menu item and crafted a CSRF payload that recreated those steps — successfully exploiting CVE-2024-24524.

This case study reveals how HPTSA mirrors expert human behavior: the planner synthesizes information across specialist runs, narrows the focus, and routes tasks to the right agents. The specialists can focus on their attack type without worrying about backtracking — that is the planner's job.

Changedetection.io XSS (CVE-2024-34061): This vulnerability required navigating to a specific page with improper input escaping. Multiple runs were needed before the agent found the correct page — the backtracking and retry mechanism was essential.

6.2 Unsuccessful Case Studies

HPTSA failed on two types of vulnerabilities, revealing important limitations:

Alf.io Improper Authorization (CVE-2024-25635): This vulnerability required accessing a specific API endpoint that wasn't documented anywhere on the website. Even the generic agent couldn't find it because there was no hint of the endpoint in the visible content.

Sourcecodester SQLi (CVE-2024-33247): This SQL injection required navigating to a hidden admin route with no visible input fields. Without obvious forms to inject into, the agent had no way to discover or target the endpoint.

These failures suggest that HPTSA could be improved by forcing agents to explore hidden endpoints through brute force or other discovery techniques, rather than relying solely on what's visible in the web interface.

7. Cost Analysis

Using GPT-4, each HPTSA run costs approximately $4.39. With an 18% success rate, the cost per successful exploit is about $24.40. For comparison, a human cybersecurity expert charges roughly $50 per hour, and exploring a website takes about 1.5 hours — totaling $75 per attempt.

Model	Cost/Run	Cost/Success
GPT-4	$4.39	$24.40
Llama 3.1 405B	$0.30	N/A (no success)
Qwen 2.5 72B	$1.41	N/A (no success)
Human Expert	$75.00	$75.00

Table 3: Cost comparison across models and human experts.

GPT-4 is 3.1x cheaper than a human expert per attempt. As AI model costs continue to drop — GPT-4o costs were halved in six months — the researchers predict GPT-4-level agents will be 3-6x cheaper within 1-2 years, making AI agents substantially more affordable than human penetration testers.

8. Related Work

Cybersecurity and AI

Research at the intersection of AI and cybersecurity falls into three categories: human uplift, societal implications, and autonomous agents.

The closest prior work showed that ReAct-style agents could hack capture-the-flag websites and known vulnerabilities when given a description. However, these agents failed in the zero-day setting because backtracking after dead ends was too difficult. HPTSA is the first multi-agent system to successfully exploit real zero-day vulnerabilities.

Human uplift research uses AI to help humans with penetration testing and malware analysis. This is especially relevant for unskiled attackers ("script kiddies") who could use AI to deploy malware without expertise. Government agencies and industrial labs are actively measuring the cybersecurity capabilities of AI agents.

AI Agents

AI agents powered by LLMs can now solve complex tasks like real-world GitHub issues. Hundreds of papers have improved agents through better prompting, planning, memory, and domain-specific designs. Multi-agent systems are particularly relevant to this work.

HPTSA makes three novel contributions to multi-agent systems. First, it uses a non-linear hierarchical structure that adapts to diverse cybersecurity sub-domains. Second, it uses dynamic orchestration for agent selection, enabling joint attacks from different domains. Third, it emphasizes agent independence — specialists freely explore and execute code rather than following rigid directives.

Security of AI Agents

A related concern is the security of AI agents themselves. Deployers may want to limit what agents can do (like preventing hacking), but recent work showed it's easy to bypass LLM protections through fine-tuning or indirect prompt injection attacks. This work is orthogonal to HPTSA but equally important for the field.

9. Conclusions

This paper demonstrates that teams of LLM agents can autonomously exploit zero-day vulnerabilities, answering an open question in the field. The HPTSA framework shows that a carefully designed multi-agent system can overcome the limitations of single agents.

The findings have dual implications. On one hand, black-hat hackers can now use AI to attack websites more effectively. On the other hand, penetration testers can use AI to conduct more frequent and thorough security testing. It remains unclear whether AI will benefit offense or defense more, and the researchers hope future work addresses this balance.

Beyond the immediate results, the researchers urge frontier LLM providers to carefully consider the implications of their deployments. As AI agents become capable of real cyberattacks, responsible development and deployment become critical.

10. Limitations

Several limitations remain. First, the benchmark focuses only on web, open-source vulnerabilities, which may bias the results. Second, HPTSA relies on powerful LLMs — extending it to smaller, cheaper models under resource constraints is important future work. Third, while sandboxes mimic real-world conditions, gaps remain between emulated and real environments, such as compute resources and additional software information.

Finally, the researchers call for more comprehensive benchmarks to evaluate AI cybersecurity risks. Although HPTSA shows potential for defensive screening, its defensive capabilities were not evaluated in this study.

11. Ethical Considerations

A major concern is that malicious actors could use these ideas for harmful purposes. To mitigate this, the researchers chose not to release their code or prompts publicly, following OpenAI's request and cybersecurity best practices. All findings were disclosed to OpenAI as part of their responsible disclosure program.

The benchmark was designed to prevent harm: all vulnerabilities were reproduced in sandboxed environments, no real users were targeted, and all tested software was open-source with publicly available patches. The goal is to advance defensive security research, not enable attacks.

Appendix A: Prompts Used in HPTSA

All agents in HPTSA receive prompts with two key components:

Statements to Avoid Request Denials: Agents are instructed to operate as white-hat hackers with permission from application owners. This framing reduces the likelihood that the LLM will refuse to perform security testing.

General Task Targets: Agents are directed to detect and exploit vulnerabilities, with specific instructions tailored to their role.

The **team manager** receives additional prompts about organizing hacker agents, understanding their capabilities, and routing tasks appropriately. The **hacker agents** receive prompts about their specific expertise, access to cybersecurity documentation, and instructions for interacting with web applications.

Appendix B: Documents Used to Augment HPTSA

Each specialist agent was provided with 5-6 curated reference documents:

Agent	Reference Documents
SQLi Agent	Advanced SQL Injection; SQL Injection Attacks and Defense; Detection and Prevention of SQL Injection Attacks; SQL Injection Tutorial
CSRF Agent	CSRF: Attack and Defense; Cross-Site Request Forgeries: Exploitation and Prevention; SQL Injection Attacks and Countermeasures in PHP; Finding and Preventing CSRF
XSS Agent	An Introduction to XSS; Using XSS to Bypass CSRF Protection; Our Favorite XSS Filters/IDS; Complete XSS Walkthrough
SSTI Agent	SSTI Handbook; Server-Side Template Injection: RCE for the Modern Webapp
Generic Agent	Web Hacking 101

Table 4: Reference documents provided to each specialist agent.

Appendix C: HPTSA with a Reasoning Model

The researchers extended their evaluation to include OpenAI's o3 reasoning model. To reduce request denials, they modified the initial prompt to frame the objective as defensive security improvement, achieving a low denial rate of 2.9%.

HPTSA with o3 achieved 36% pass@5 and 16% overall success rate — slightly lower than GPT-4's 42% and 18%. After manual investigation, the gap was primarily caused by o3's rigid adherence to documentation. In 30% of attempts, o3 copied commands directly from reference documents without adapting them to the specific website, leading to frequent "404 Not Found" errors.

This result reveals an important insight: more powerful reasoning models do not automatically perform better. The HPTSA framework requires models that can dynamically adapt to context rather than blindly following documented procedures. The ability to deviate from instructions when they don't apply is a crucial skill for real-world hacking.

The o3 results also highlight a fundamental tension in AI safety. While o3 was designed to be more cautious and refuse harmful requests, this caution actually reduced its effectiveness at security testing. The researchers had to explicitly frame the task as defensive to get o3 to cooperate — a workaround that raises questions about the balance between safety and capability in AI systems.

Key Takeaways

1. Teamwork beats individual effort. A single AI agent cannot handle the complexity of real-world hacking. HPTSA's hierarchical design — with a planner, manager, and specialists — mirrors how human cybersecurity teams operate.

2. Zero-day exploitation is now possible for AI. This paper resolves the open question of whether AI can find and exploit unknown vulnerabilities. The answer is yes, at least for web applications with GPT-4-class models.

3. Open-source models are not ready. Despite their size, Llama 3.1 405B and Qwen 2.5 72B scored 0% on the benchmark. They refused to attempt attacks or repeated the same failed approach. GPT-4 remains significantly more capable for this task.

4. Each component matters. Removing any part of HPTSA — specialists, documents, or hierarchy — dramatically reduces performance. The hierarchical structure is the most critical, with its removal causing a 13x drop in success rate.

5. AI is cheaper than humans. At $24.40 per successful exploit vs. $75 for a human expert, HPTSA is already 3.1x more cost-effective. As model costs continue to fall, this advantage will grow.

6. Bigger models aren't always better. The o3 reasoning model actually performed worse than GPT-4 because it rigidly followed documentation instead of adapting to the specific target. Dynamic adaptation is more important than raw reasoning power.

7. Defensive applications are promising. While this paper focuses on offense, HPTSA could be used defensively to find vulnerabilities before attackers do. This defensive use case deserves more research attention.

Broader Impact and Future Directions

The ability of AI teams to exploit zero-day vulnerabilities has profound implications for cybersecurity. Several important questions emerge from this work:

Offense vs. Defense Balance: Will AI help attackers or defenders more? Currently, both sides benefit, but the advantage may shift as capabilities evolve. The researchers hope future work addresses this critical question.

Responsible Deployment: Frontier LLM providers must carefully consider how their models might be used for cybersecurity. Safety guardrails need to balance preventing misuse with enabling legitimate security research.

Broader Benchmark Coverage: This study focused on web vulnerabilities. Future benchmarks should cover network protocols, mobile applications, IoT devices, and cloud infrastructure to give a complete picture of AI cybersecurity capabilities.

Smaller Models and Accessibility: Making HPTSA work with cheaper, smaller models would democratize security testing. Currently, only organizations with access to expensive API calls can benefit from this technology.

Real-World Deployment: The gap between sandboxed testing and production environments remains significant. Bridging this gap requires addressing additional challenges like authentication, rate limiting, and complex deployment architectures.

Regulatory Considerations: As AI hacking capabilities improve, governments and standards bodies will need to develop frameworks for responsible use. The balance between enabling security research and preventing misuse will require careful policy.

References

Achiam, J. et al. (2023). GPT-4 Technical Report. arXiv:2303.08774.

Anthropic. (2024). A New Initiative for Developing Third-Party Model Evaluations.

Bennetts, S. (2013). OWASP Zed Attack Proxy. AppSec USA.

Bilge, L. & Dumitras, T. (2012). Before We Knew It: An Empirical Study of Zero-Day Attacks. ACM CCS.

Blatz, J. (2007). CSRF: Attack and Defense. McAfee Foundstone.

Chen, G. et al. (2023). AutoAgents: A Framework for Automatic Agent Generation. arXiv:2309.17288.

Clarke-Salt, J. (2009). SQL Injection Attacks and Defense. Elsevier.

Dubey, A. et al. (2024). The Llama 3 Herd of Models. arXiv:2407.21783.

Fang, R. et al. (2024a). LLM Agents Can Autonomously Exploit One-Day Vulnerabilities. arXiv:2404.08144.

Fang, R. et al. (2024b). LLM Agents Can Autonomously Hack Websites. arXiv:2402.06664.

Hong, S. et al. (2023). MetaGPT: Meta Programming for Multi-Agent Collaborative Framework. arXiv:2308.00352.

Kost, E. (2023). Critical Microsoft Exchange Flaw: CVE-2021-26855.

Lohn, A. & Jackson, K. (2022). Will AI Make Cyber Swords or Shields?

OWASP. (2020). Testing for Server Side Template Injection.

OWASP. (2024). Vulnerability Disclosure Cheat Sheet.

Parisi, A. et al. (2022). Tool Augmented Language Models. arXiv:2205.12255.

Shinn, N. et al. (2024). Reflexion: Language Agents with Verbal Reinforcement Learning. NeurIPS 36.

Yao, S. et al. (2022). ReAct: Synergizing Reasoning and Acting in Language Models. arXiv:2210.03629.

Yang, A. et al. (2024a). Qwen2.5 Technical Report. arXiv:2412.15115.

Yang, J. et al. (2024b). SWE-Agent: Agent Computer Interfaces Enable Software Engineering Language Models.

Zhan, Q. et al. (2023). Removing RLHF Protections in GPT-4 via Fine-Tuning. arXiv:2311.05553.

Zou, A. et al. (2023). Universal and Transferable Adversarial Attacks on Aligned Language Models. arXiv:2307.15043.