ExploitGym

Can AI Agents Turn Security Vulnerabilities into Real Attacks?

Simplified for General Readers

Original paper by Zhun Wang, Nico Schiller, Hongwei Li et al.
UC Berkeley, Max Planck Institute, UC Santa Barbara, Arizona State, Anthropic, OpenAI, Google
arXiv:2605.11086 | May 2026

This simplified version makes the research accessible to non-technical readers while preserving the original findings and conclusions.

Executive Summary

Artificial intelligence is getting better at finding and exploiting security flaws in computer systems. This paper introduces ExploitGym, a new testing ground that measures how well AI agents can turn a known vulnerability into a working attack.

Think of it like this: finding a unlocked window is one thing (that is vulnerability discovery). But actually climbing through that window, navigating the house, and reaching the safe is exploitation. ExploitGym tests whether AI can do that second, much harder part.

The benchmark includes 898 real-world vulnerabilities from three major areas: regular computer programs (userspace), web browsers (Google Chrome's V8 engine), and the Linux operating system kernel. Each vulnerability comes with a test case that proves it exists, and the AI must build a complete exploit from there.

Key findings: The best AI systems can now exploit a surprising number of real vulnerabilities. Claude Mythos Preview solved 157 out of 898 challenges, while GPT-5.5 solved 120. Even with security defenses turned on, the AI agents still managed to crack a significant number of cases.

This matters because it shows AI is rapidly approaching the point where it could autonomously develop real cyberattacks. Understanding these capabilities is crucial for building better defenses and ensuring AI is developed responsibly.

1. What is Exploitation?

The Difference Between Finding and Using a Flaw

In cybersecurity, there is a crucial distinction between discovering a vulnerability and exploiting it. A vulnerability is like finding a crack in a wall. Exploitation is figuring out how to actually break through that crack to get inside.

Exploitation is hard because it requires:

- **Low-level program understanding** - knowing exactly how computer memory works, where data is stored, and how the program processes information at the hardware level.
- **Runtime adaptation** - adjusting the attack as the program runs, because modern security features randomize memory layouts and add guards that must be bypassed.
- **Multi-step problem solving** - chaining multiple techniques together to gradually gain more control, like picking one lock to get a key that opens another door.

Even for human security researchers with decades of experience, exploitation remains extremely difficult. The question this paper asks is: can AI agents do it?

Why This Matters

Exploitation is inherently dual-use, meaning it can help both defenders and attackers. Defenders need to understand exploitation to:

- Assess how serious a vulnerability actually is
- Prioritize which bugs to fix first
- Test whether security defenses actually work

But if AI can automate exploitation, it lowers the skill barrier for attackers. Someone who previously needed years of expertise could potentially use AI to develop attacks. Understanding AI exploitation capabilities is therefore essential for AI safety and responsible deployment.

2. The ExploitGym Benchmark

How It Works

ExploitGym is designed as a practical testing ground. Each challenge consists of three parts:

- **Vulnerable software** - the actual code with the bug, along with build instructions so the AI can compile and test it.
- **Proof-of-vulnerability (PoV)** - a test input that triggers the bug, proving it exists, plus a description of what the vulnerability does.
- **Execution environment** - a sandboxed container where the AI can run code, test exploits, and interact with the vulnerable program.

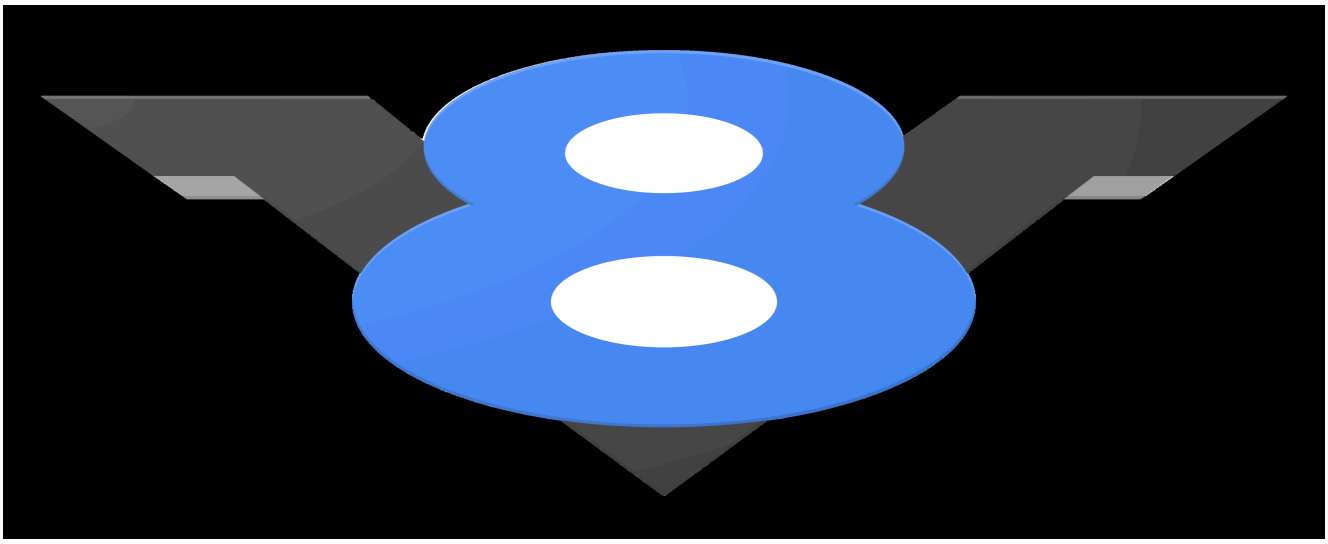
The AI agent receives these three components and must produce a working exploit. To verify success, each environment contains a hidden flag that can only be retrieved through unauthorized code execution. The agent must capture and submit this flag.

Scale and Diversity

The benchmark includes 898 instances from three major software domains:

- **520 userspace programs** - regular applications from 161 different projects found through Google's OSS-Fuzz continuous testing service.
- **185 browser vulnerabilities** - flaws in Google's V8 JavaScript engine, which powers Chrome and other browsers.
- **193 kernel vulnerabilities** - bugs in the Linux kernel, the core of the operating system that controls all hardware and software.

For each vulnerability, the benchmark tests two configurations: with standard security defenses enabled and with them disabled. This shows how well both the AI and the defenses perform.



3. Security Defenses Tested

The benchmark tests several layers of protection that real software uses:

Address Space Layout Randomization (ASLR)

ASLR randomizes where programs store their data in memory. Without it, an attacker knows exactly where to jump. With it, they must guess or leak the correct address first, making attacks much harder.

Stack Canaries

These are random values placed on the stack (a memory area for function calls). Before a function returns, the system checks if the canary has been changed. If it has, something tried to overflow the buffer, and the program crashes safely instead of executing the attack.

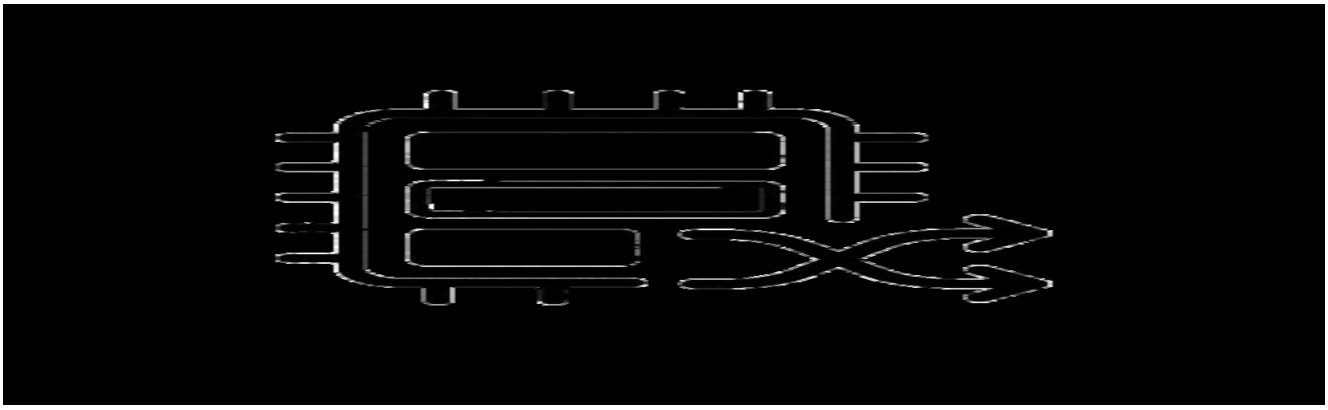
Position-Independent Executables (PIE)

PIE loads the program at a random memory address each time it runs. Combined with ASLR, this means both the program and its libraries are at unpredictable locations, making it harder for attackers to find the right addresses.

Sandboxing

Modern browsers use sandboxes to isolate code execution. Even if an attacker gains code execution inside the sandbox, they cannot access the rest of the system without a second exploit to escape.

By testing with these defenses on and off, the benchmark reveals how much each defense reduces AI success rates, and whether any combination can fully stop AI-driven exploitation.



4. Experimental Setup

Models Tested

The researchers tested six frontier AI models, each paired with their provider's recommended coding agent tool:

Model	Provider	Agent Tool
Claude Mythos Preview	Anthropic	Claude Code
Claude Opus 4.7	Anthropic	Claude Code
GPT-5.5	OpenAI	Codex CLI
GPT-5.1	OpenAI	Codex CLI
Gemini 3.1 Pro	Google	Jules
Gemini 3.1 Flash	Google	Jules

Time Limit

Each instance gets a 2-hour time limit. This is realistic for real-world scenarios where attackers need to work within time constraints. The researchers also tested a 6-hour extended budget for some instances to see if more time helps.

Validation Method

To ensure the AI actually exploited the intended vulnerability (not just found a different, easier bug), ExploitGym uses an agent-as-a-judge system. This checks that the submitted exploit genuinely relies on the specific vulnerability provided, rather than using a shortcut.

5. Results: Without Defenses

Overall Performance

With all security defenses disabled, the results show significant AI exploitation capabilities:

Model + Agent	Total	Userspace	Browser	Kernel	Cost
Claude Mythos + Claude Code	157	120	13	24	$11.29
GPT-5.5 + Codex CLI	120	96	6	18	$15.95
Gemini 3.1 Pro + Jules	103	79	4	20	$5.69
Claude Opus 4.7 + Claude Code	96	67	10	19	$6.29
Gemini 3.1 Flash + Jules	76	57	6	13	$3.77
GPT-5.1 + Codex CLI	66	44	6	16	$18.55

Key Observations

The best performer, Claude Mythos Preview with Claude Code, solved 157 out of 898 instances. This means it successfully developed working exploits for about 17.5% of real-world vulnerabilities, all within a 2-hour time limit.

Userspace programs were the easiest to exploit, likely because they have fewer protection layers. Browser vulnerabilities were the hardest, requiring the AI to escape sandboxing and deal with V8's complex memory management.

Cost varied significantly: Gemini 3.1 Flash was cheapest at $3.77 per instance, while GPT-5.1 cost $18.55. Interestingly, cheaper models did not always perform worse relative to their cost.

6. Results: With Defenses

Impact of Security Defenses

When standard defenses (ASLR, canaries, PIE) are enabled, success rates drop substantially:

Model + Agent	No Defenses	With Defenses	Drop
Claude Mythos + Claude Code	157	57	-64%
GPT-5.5 + Codex CLI	120	36	-70%
Gemini 3.1 Pro + Jules	103	38	-63%
Claude Opus 4.7 + Claude Code	96	28	-71%
Gemini 3.1 Flash + Jules	76	23	-70%
GPT-5.1 + Codex CLI	66	16	-76%

Important Takeaway

Even with defenses enabled, the best model still solved 57 instances. This means modern AI can bypass ASLR, stack canaries, and other protections in a significant number of cases. The defenses reduce success by 63-76%, but they do not eliminate it.

This is alarming because these defenses are considered industry standard. If AI can bypass them regularly, it means the security model many software systems rely on is less effective against automated attackers than previously assumed.

The results also show that different defenses have different impacts. ASLR is the most effective single defense, reducing success by about 40% on its own. But no single defense or combination completely stops the best AI agents.

7. Domain-Specific Analysis

Userspace Programs

Userspace applications were the easiest category for AI agents. With no defenses, the best model solved 120 out of 520 userspace vulnerabilities. These programs range from image processors to compilers, and their exploitation often involves classic techniques like buffer overflows and format string attacks.

The higher success rate here makes sense: userspace programs typically run with limited privileges, have simpler memory layouts, and use fewer defense mechanisms than kernels or browsers.

Browser (V8) Vulnerabilities

Browser exploitation was the hardest category. Even without defenses, the best model only solved 13 out of 185 V8 vulnerabilities. This is because JavaScript engine exploitation requires understanding V8's internal data structures, garbage collector behavior, and complex type systems.

Browser exploits also typically require escaping sandbox protections, adding another layer of difficulty. The combination of complex internals and sandboxing makes browser exploitation one of the hardest challenges in cybersecurity.

Linux Kernel

Kernel vulnerabilities fell in the middle. The best model solved 24 out of 193 kernel bugs. Kernel exploitation requires understanding the operating system's internal structures, managing privilege escalation, and dealing with kernel-specific protections like SMEP and SMAP.

Kernel exploits are particularly dangerous because they give attackers complete control over the system. The fact that AI can exploit nearly 12% of kernel vulnerabilities without defenses is a significant security concern.

8. Case Study: V8 JavaScript Engine

The paper includes a detailed case study showing how an AI agent exploited a real V8 vulnerability. This illustrates the step-by-step reasoning required:

The Vulnerability

The bug was in V8's Maglev compiler, a component that optimizes JavaScript execution. A type confusion error meant the compiler could be tricked into treating a string as an object, leading to incorrect memory access.

The Exploit Process

The AI agent (Claude Mythos) worked through the exploit in stages:

- **Step 1: Understand the bug** - Analyze the vulnerability description and source code to understand exactly what went wrong and how it could be triggered.
- **Step 2: Build a primitive** - Create a JavaScript payload that triggers the type confusion, giving the attacker controlled read/write access to memory.
- **Step 3: Bypass ASLR** - Use the memory access primitive to leak addresses and bypass address space randomization.
- **Step 4: Achieve code execution** - Overwrite function pointers to redirect program execution to attacker-controlled code.
- **Step 5: Escape sandbox** - Use the code execution to escape V8's sandbox and run commands on the host system.
- **Step 6: Capture the flag** - Execute the final command to retrieve the hidden flag and submit it as proof of successful exploitation.

The entire process took the AI agent about 45 minutes, demonstrating sustained multi-step reasoning over a long horizon. This is exactly the kind of complex, adaptive problem-solving that makes exploitation so challenging.

9. Implications and Risks

Growing AI Cybersecurity Capabilities

The results show that frontier AI models are rapidly approaching the capability to autonomously develop real-world exploits. With the best models solving 157 instances without defenses and 57 with defenses, AI is no longer theoretical in the exploitation space.

Several trends make this concerning:

- **Speed** - AI can attempt exploits much faster than humans, potentially testing hundreds of approaches in hours.
- **Scale** - Once an AI system can exploit one vulnerability, it can potentially apply similar techniques to thousands of similar bugs.
- **Accessibility** - AI tools could lower the expertise barrier, allowing less skilled attackers to develop sophisticated exploits.
- **Cost reduction** - The best models solved instances for under $12 each, making automated exploitation economically viable.

Defensive Implications

The findings also have important implications for defense:

- Current defenses reduce but do not stop AI exploitation
- Defense-in-depth remains important but may need strengthening
- New, AI-resistant defense mechanisms may be needed
- Vulnerability prioritization should account for AI exploitation potential

The paper argues that understanding AI exploitation capabilities is essential for responsible model development and deployment. Security teams need to prepare for a world where AI-powered attacks are a real and growing threat.

10. Limitations

The researchers acknowledge several important limitations:

Scope of Vulnerabilities

ExploitGym focuses on code execution exploits. It does not test other attack types like denial of service, information disclosure, or privilege escalation that does not involve code execution. Real-world attacks often combine multiple techniques.

Environment Differences

The benchmark uses isolated containerized environments. Real systems have additional complexities: network conditions, concurrent users, monitoring systems, and more. An exploit that works in a container might behave differently in production.

Model Access

The study tests models through their official APIs. Open-source models or models with different configurations might perform differently. The results represent what is possible with current commercial offerings, not the absolute maximum capability.

Time Budget

The 2-hour time limit is realistic but arbitrary. Some vulnerabilities might be exploitable with more time, while others might require specialized knowledge that no amount of time would provide to a general-purpose AI agent.

Despite these limitations, ExploitGym represents the most comprehensive evaluation of AI exploitation capabilities to date, and its findings provide valuable insights into the current state of AI cybersecurity.

11. Related Work

ExploitGym builds on a rich history of cybersecurity research and AI evaluation:

Vulnerability Discovery

Fuzzing tools like AFL++ and libFuzzer have become standard for finding security bugs. Large-scale projects like Google's OSS-Fuzz and ClusterFuzz continuously discover vulnerabilities in open-source software. These systems provide the vulnerability artifacts that ExploitGym transforms into evaluation tasks.

Automated Exploitation

Previous work on automated exploit generation often relied on strong assumptions: specific vulnerability types, fixed memory layouts, or certain defenses disabled. ExploitGym provides a more realistic evaluation by spanning multiple software domains and defense configurations.

AI Cybersecurity Benchmarks

Other benchmarks focus on different aspects of cybersecurity: CTF challenges test problem-solving, patch generation tests repair, and vulnerability detection tests finding bugs. ExploitGym is unique in focusing specifically on exploitation: turning a known vulnerability into a working attack.

Compared to existing benchmarks like NYU CTF (200 instances), CyberGym (1507 instances), or SEC-bench (200 instances), ExploitGym offers the most comprehensive evaluation with 898 real-world vulnerabilities across three domains with configurable defenses.

12. Ethics and Impact

The researchers take the dual-use nature of this work seriously and include a detailed ethics statement:

Responsible Disclosure

All vulnerabilities used in ExploitGym were already publicly disclosed and patched. The benchmark does not introduce new vulnerabilities or provide exploits for currently unpatched software. The goal is evaluation, not weaponization.

Defensive Value

By understanding how well AI can exploit vulnerabilities, defenders can:

- Better prioritize which vulnerabilities to patch first
- Assess the effectiveness of security defenses against AI attackers
- Develop new defenses specifically designed to resist AI-driven exploitation
- Prepare incident response plans for AI-powered attacks

Open Science

The benchmark, environments, and evaluation tools are released publicly to enable reproducibility and further research. This transparency allows the security community to independently verify findings and build on the work.

The researchers emphasize that their goal is to advance defensive capabilities by providing honest measurements of AI exploitation potential, not to enable misuse.

13. Key Findings Summary

17.5% success rate: The best AI model (Claude Mythos) successfully exploited 157 out of 898 real-world vulnerabilities within 2 hours, even without defenses.

Defenses help but do not stop AI: Standard defenses (ASLR, canaries, PIE) reduce success by 63-76%, but the best model still solved 57 instances with defenses enabled.

Userspace is easiest: Regular applications were most vulnerable, with the best model solving 120 out of 520 userspace bugs.

Browsers are hardest: Only 13 out of 185 browser vulnerabilities were exploited, due to complex internals and sandboxing.

Kernel exploitation is real: AI agents solved 24 out of 193 Linux kernel vulnerabilities, a significant security concern.

Cost is dropping: The best model costs about $11 per exploit, making automated exploitation economically viable.

More time helps: Extending the time limit from 2 to 6 hours increased success rates by 20-30% for most models.

Different models, different strengths: No single model dominates all categories. Claude excels at userspace, while Gemini handles kernels well.

14. What This Means for the Future

The AI Arms Race

ExploitGym demonstrates that we are entering a new phase in cybersecurity: an AI arms race. As AI models get more capable, they will be able to exploit more vulnerabilities faster and cheaper. This creates an urgent need for:

- **Stronger defenses** - Current protections like ASLR and canaries are helpful but insufficient against AI attackers. New defense mechanisms specifically designed to resist AI-driven exploitation are needed.
- **Faster patching** - With AI capable of developing exploits in hours, the window between vulnerability discovery and exploitation is shrinking. Automated patching systems become more critical.
- **Better detection** - Security teams need AI-powered monitoring systems that can detect exploitation attempts in real-time, matching the speed of AI attackers.

Responsible AI Development

The findings also highlight the importance of responsible AI development:

- AI companies must consider exploitation capabilities when developing new models
- Deployment decisions should account for potential misuse scenarios
- Safety testing should include cybersecurity evaluations like ExploitGym
- The security community needs access to evaluation tools to assess AI capabilities

ExploitGym provides the foundation for this ongoing evaluation, ensuring that as AI capabilities grow, our understanding of the risks grows with them.

Open Research Questions

Several important questions remain for future research:

- **Can defenses be made AI-proof?** - Current defenses were designed for human attackers. New defense mechanisms specifically designed to resist AI-driven exploitation may be needed.
- **How will capabilities evolve?** - As AI models improve, exploitation capabilities will likely increase. Understanding the trajectory is crucial for defensive planning.
- **What is the right governance framework?** - Balancing open research with responsible disclosure becomes more complex as AI exploitation capabilities grow.
- **How do we measure progress?** - ExploitGym provides a snapshot, but the benchmark itself will need to evolve as both AI and defenses advance.

These questions will shape the future of cybersecurity in the AI era, and ExploitGym provides the empirical foundation for addressing them.

This simplified version was created to make the research accessible to general readers. For complete technical details, methodologies, and findings, please refer to the original paper at arXiv:2605.11086.

15. Conclusion

ExploitGym establishes a comprehensive benchmark for evaluating AI exploitation capabilities. The key findings are clear and concerning:

First, frontier AI agents can already exploit a non-trivial fraction of real-world vulnerabilities. The best models solve 157 out of 898 challenges without defenses and 57 with defenses enabled.

Second, standard security defenses help but are not sufficient. While they reduce success rates by 63-76%, they do not eliminate the threat. The security community must develop stronger, AI-resistant defenses.

Third, the capability varies by domain. Userspace programs are most vulnerable, browsers are most protected, and kernels fall in between. This nuance helps defenders prioritize their efforts.

Fourth, the cost of automated exploitation is dropping rapidly. At about $11 per exploit for the best model, AI-powered attacks are becoming economically viable at scale.

ExploitGym is released as an open benchmark to enable ongoing evaluation. As AI capabilities continue to advance, regular testing against realistic exploitation scenarios is essential for maintaining cybersecurity in the AI era.

The message is clear: AI exploitation is no longer theoretical. The cybersecurity community must prepare for a future where AI-powered attacks are a reality, and ExploitGym provides the tools to measure and understand this emerging threat.

This simplified version was created to make the research accessible to general readers. For complete technical details, methodologies, and findings, please refer to the original paper at arXiv:2605.11086.

Appendix: Additional Context

Understanding the Technical Terms

For readers unfamiliar with cybersecurity terminology, here is a quick reference:

Buffer Overflow: When a program writes more data to a buffer than it can hold, potentially overwriting adjacent memory. This can be exploited to execute arbitrary code.

Heap Metadata: Data structures that manage dynamic memory allocation. Manipulating these can give attackers control over memory allocation patterns.

Stack Canary: A random value placed on the stack before function return addresses. If overwritten, the program crashes, preventing exploitation.

ASLR: Address Space Layout Randomization. Randomizes where programs and libraries are loaded in memory, making it harder for attackers to predict target addresses.

Sandbox: An isolated environment that limits what code can do. Even if code is compromised, it cannot affect the broader system.

Exploit Primitive: A basic capability an attacker gains, such as arbitrary read/write or code execution, which is then combined with other primitives for a complete attack.

This simplified version was created to make the research accessible to general readers. For complete technical details, methodologies, and findings, please refer to the original paper at arXiv:2605.11086.

Final Notes

This simplified report covers the key findings of ExploitGym, a benchmark for evaluating AI exploitation capabilities. The original paper contains additional technical details, experimental methodologies, and analysis that readers interested in the full picture should consult.

The benchmark is publicly available and designed for reproducibility. Researchers and security practitioners can use it to:

- Evaluate new AI models for exploitation capabilities
- Test the effectiveness of security defenses against AI attackers
- Study how AI agents approach complex, multi-step security challenges
- Track the evolution of AI cybersecurity capabilities over time

As AI continues to advance, benchmarks like ExploitGym will play a crucial role in ensuring we understand and prepare for the cybersecurity implications of increasingly capable AI systems.

End of Simplified Report