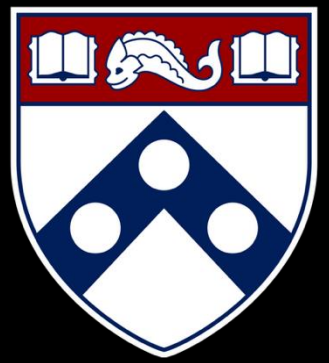
Secure Systems Engineering and Management

A Data-driven Approach

Michael Hicks



How GenAI Reshapes the Cybersecurity Landscape

UPenn CIS 7000-003
Spring 2026

GenAI is a general-purpose **capability amplifier**

for both attack and (hopefully) defense

Amplifying Attacks

A financially motivated attacker, today:

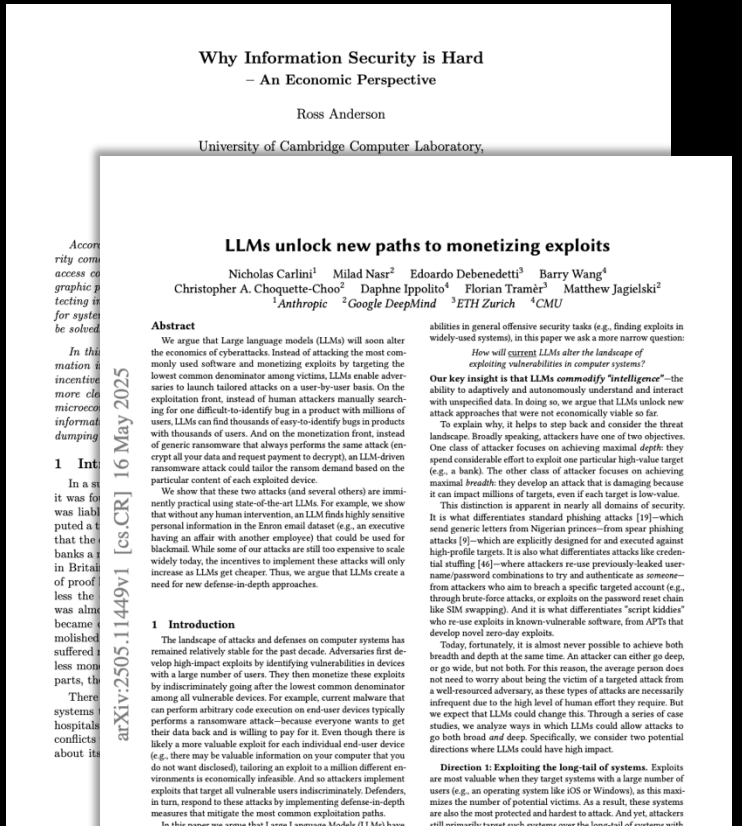
- What platform should I target when developing a RCE exploit?
  - Answer #1: A highly used platform. The RCE is hard to develop, but I do it I can monetize it via ransomware on lots of users.
  - Answer #2: Many barely used platforms. The RCEs are easy to develop, which makes up them having fewer users.
- Can I boost expected value by targeting attacks, e.g., not just doing generic ransomware?
  - Each user surely has things they are willing to pay more for!
- In practice, it's always answer #1 and 'no': even easy-to-develop RCEs have a high cost, and targeting attacks makes that cost higher.

GenAI changes the incentives

LLMs commodify intelligence.
This changes:

- **Who** can execute attacks
- At what **cost**
- At what **scale**

LLMs are reshaping the **economics** of security



Why Information Security is Hard
– An Economic Perspective

Ross Anderson

University of Cambridge Computer Laboratory,

LLMs unlock new paths to monetizing exploits

Nicholas Carlini[1] Milad Nasr[2] Edoardo Debenedetti[3] Barry Wang[4]
Christopher A. Choquette-Choo[2] Daphne Ippolito[4] Florian Tramèr[3] Matthew Jagielski[2]
[1]Anthropic [2]Google DeepMind [3]ETH Zurich [4]CMU

arXiv:2505.11449v1 [cs.CR] 16 May 2025

Recall: Security Economics

Attacks occur when **expected profit exceeds cost**

$$\text{value} = (\text{profit per exploit}) \times (\#\text{impacted}) - (\text{cost to find \& exploit})$$

LLMs apply pressure on all three levers

Lever 1: Profit Per Exploit

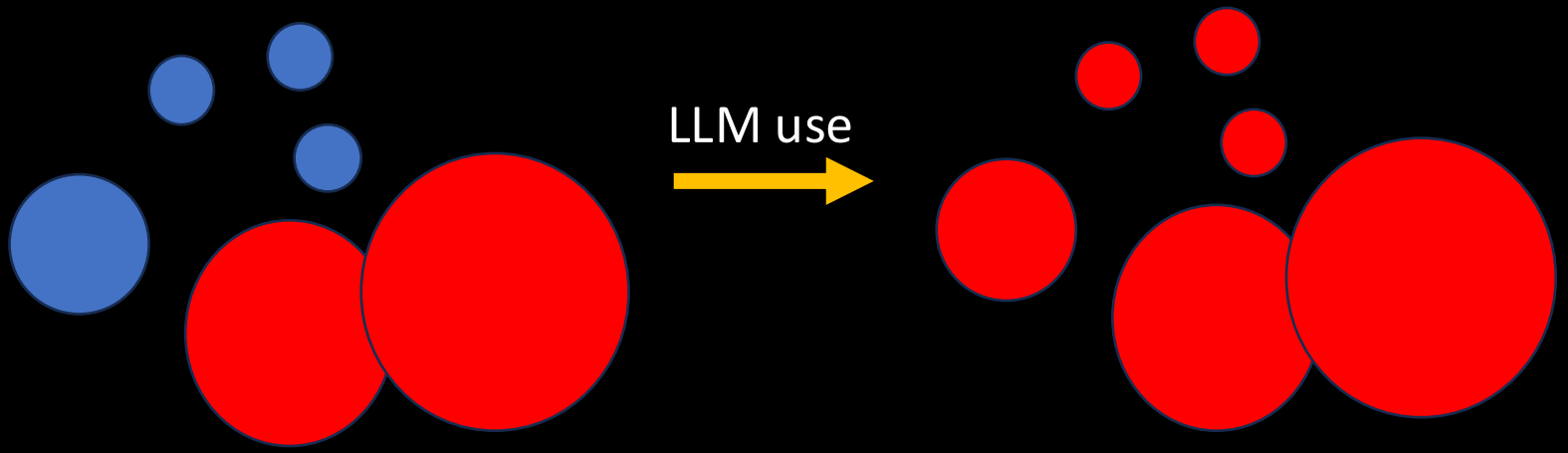
LLMs enable **personalized monetization** rather than one-size-fits-all

Generic ransomware ⇒ **targeted blackmail**

Lever 2: Number of Targets

LLMs reduce **marginal per-target cost**

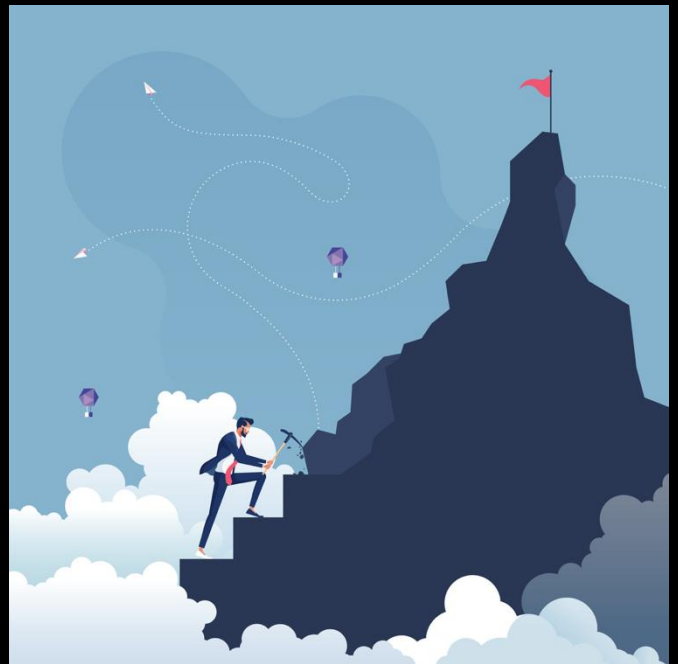
Previously uneconomical targets come into range



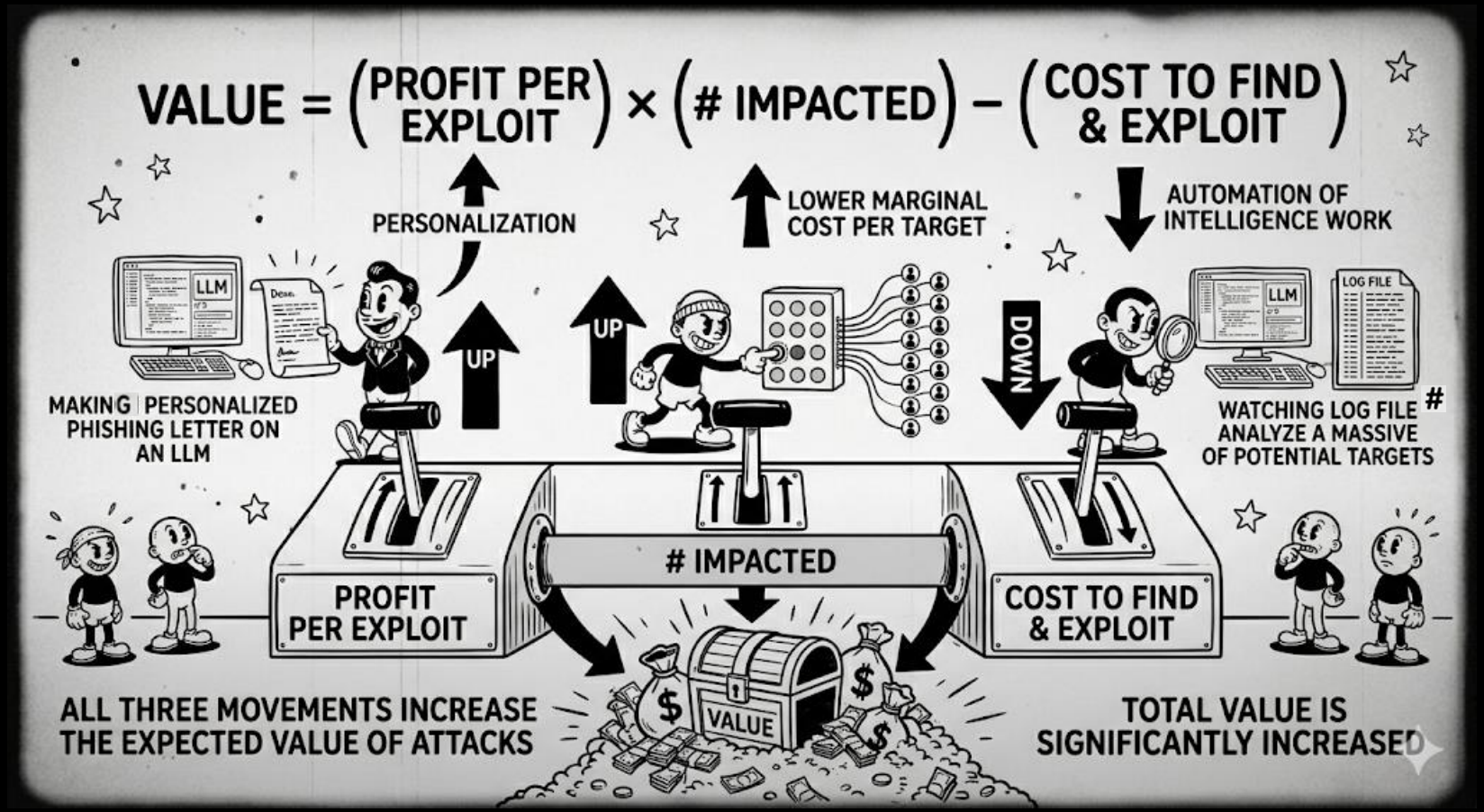
Lever 3: Resources to Find and Exploit

LLMs automate the “intelligence work” that required **skilled human time**

- Reconnaissance
- Vulnerability analysis
- Exploit development
- Social engineering crafting



Profit per exploit UP, number of targets UP, cost DOWN



The Breadth vs. Depth Trade-Off

Historically, attackers chose:

- **Wide**: generic attacks, many targets (script kiddies)
- **Deep**: tailored attacks, high-value single target (APTs)

LLMs may **collapse this choice**

Example: One-upping Ransomware

Traditional ransomware has a distinctive forensic footprint:

- Mass disk writes
- Ransom note
- Encrypted files
- Detectable via file-system monitoring

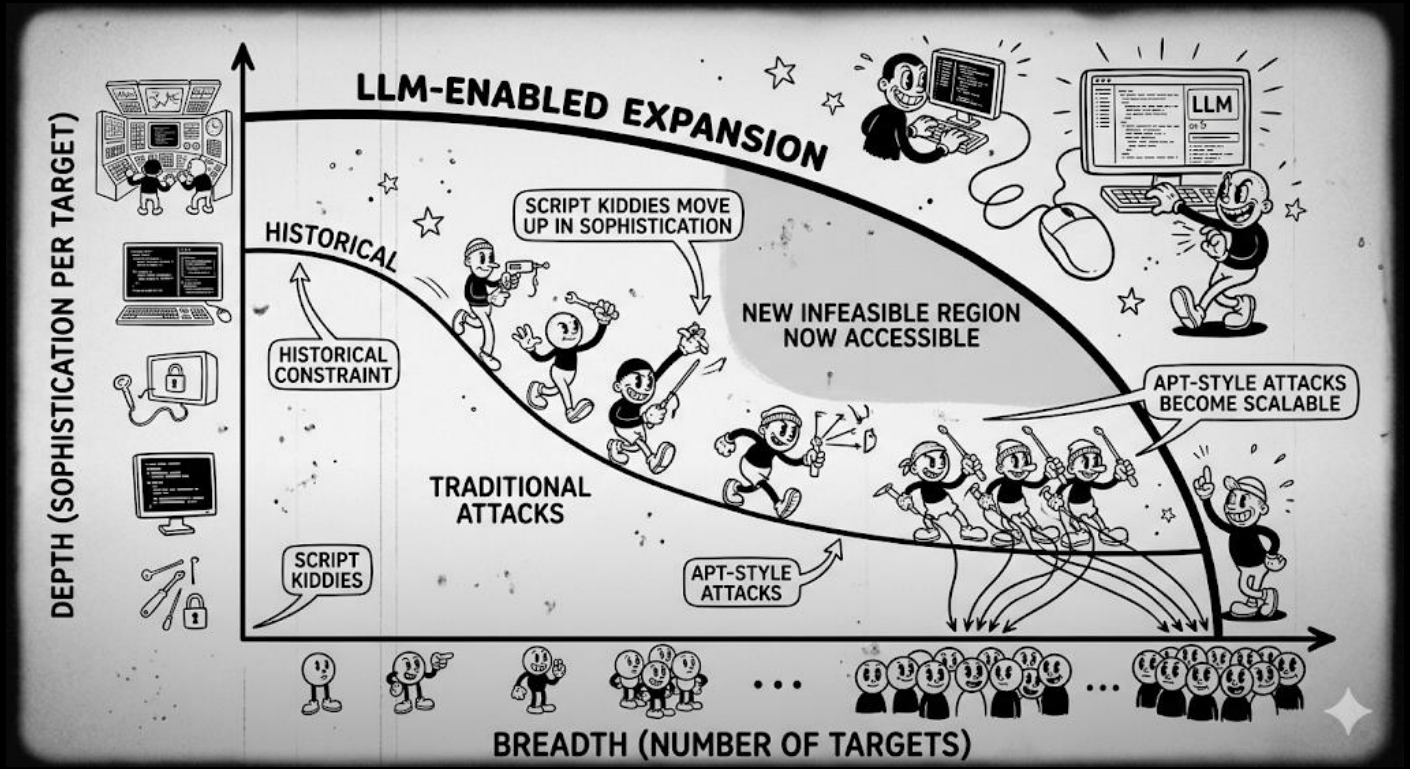
LLM-Powered Intelligent Exfiltration

The LLM runs on the **attacker's infrastructure**

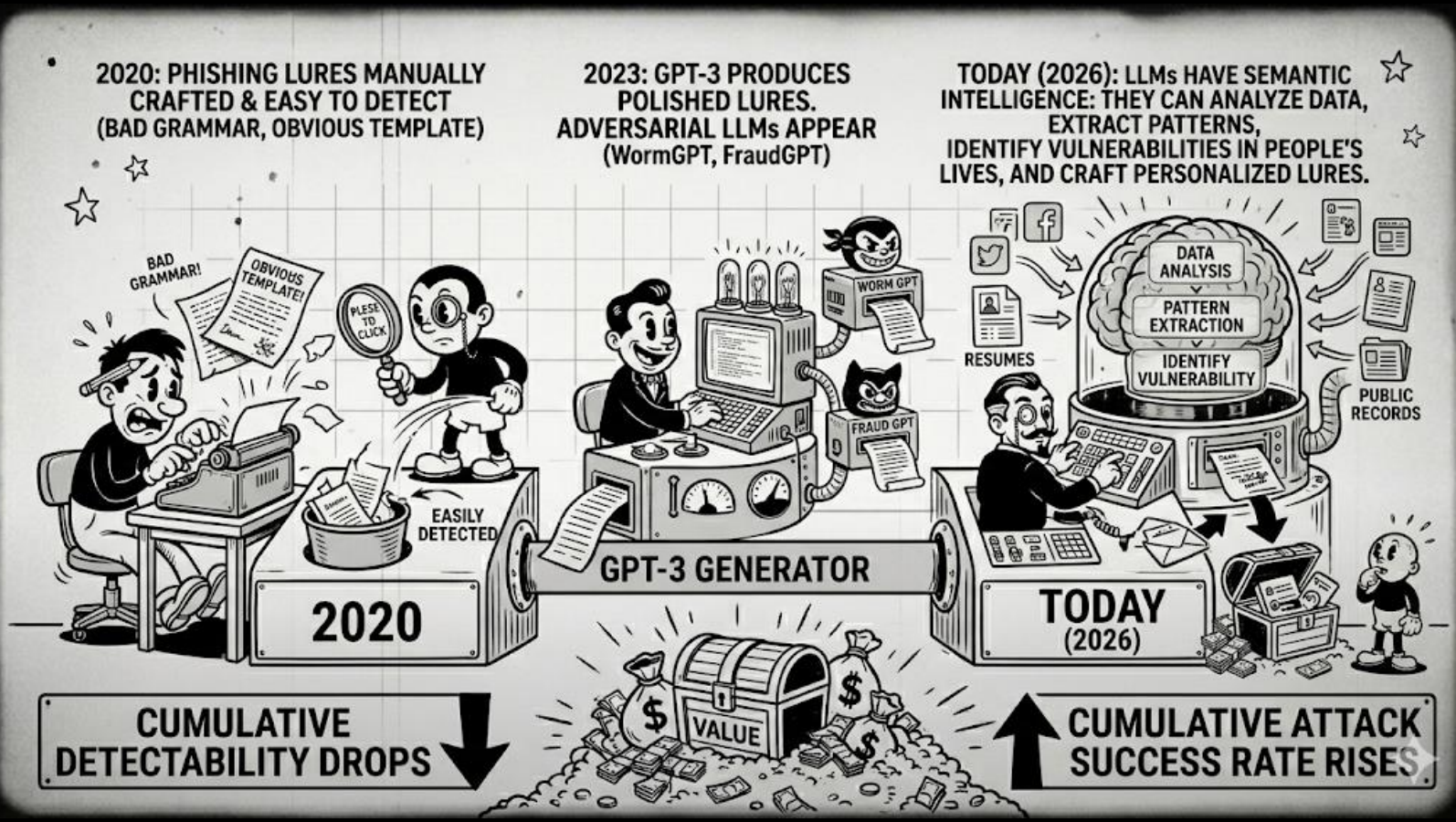
A lightweight agent on the victim device just reads and sends files

Ransomware	Intelligent Exfiltration
Files are **modified/encrypted**	Files are **read, not modified**
Large disk write footprint	No disk writes
Bulk data theft	**Selective** (only what's valuable)
Victim knows immediately	Victim may not know **until blackmail demand**

The Effect of LLMs

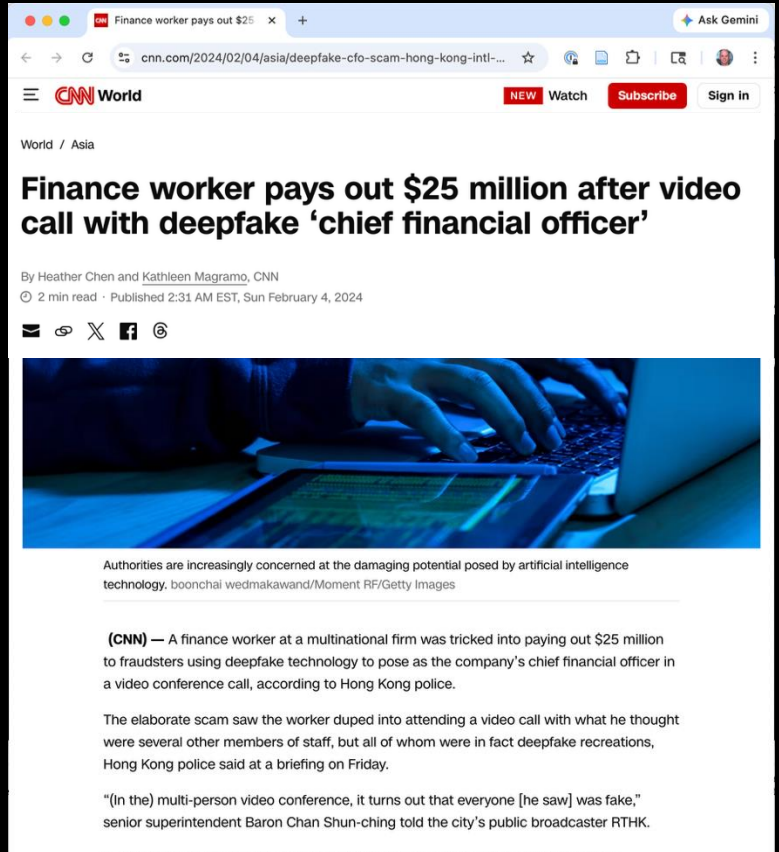


Exploiting Humans: Social Engineering


2020: PHISHING LURES MANUALLY CRAFTED & EASY TO DETECT
(BAD GRAMMAR, OBVIOUS TEMPLATE)
2023: GPT-3 PRODUCES POLISHED LURES. ADVERSARIAL LLMs APPEAR
(WormGPT, FraudGPT)
TODAY (2026): LLMs HAVE SEMANTIC INTELLIGENCE: THEY CAN ANALYZE DATA, EXTRACT PATTERNS, IDENTIFY VULNERABILITIES IN PEOPLE'S LIVES, AND CRAFT PERSONALIZED LURES.
BAD GRAMMAR!
OBVIOUS TEMPLATE!
PLESE TO CLICK
EASILY DETECTED
2020
WORM GPT
FRAUD GPT
GPT-3 GENERATOR
DATA ANALYSIS
PATTERN EXTRACTION
IDENTIFY VULNERABILITY
RESUMES
PUBLIC RECORDS
TODAY (2026)
VALUE
CUMULATIVE DETECTABILITY DROPS
CUMULATIVE ATTACK SUCCESS RATE RISES

Case Study: Deepfake Video Call

Hong Kong, 2024: A finance worker wired **$25 million** after a video call populated entirely by **deepfakes of colleagues**, including the CFO



Finance worker pays out $25

Ask Gemini

cnn.com/2024/02/04/asia/deepfake-cfo-scam-hong-kong-intl-...

CNN World

NEW Watch

Subscribe

Sign in

World / Asia

Finance worker pays out $25 million after video call with deepfake 'chief financial officer'

By Heather Chen and Kathleen Magramo, CNN

2 min read · Published 2:31 AM EST, Sun February 4, 2024

Authorities are increasingly concerned at the damaging potential posed by artificial intelligence technology. boonchai wedmakawand/Moment RF/Getty Images

(CNN) — A finance worker at a multinational firm was tricked into paying out $25 million to fraudsters using deepfake technology to pose as the company's chief financial officer in a video conference call, according to Hong Kong police.

The elaborate scam saw the worker duped into attending a video call with what he thought were several other members of staff, but all of whom were in fact deepfake recreations, Hong Kong police said at a briefing on Friday.

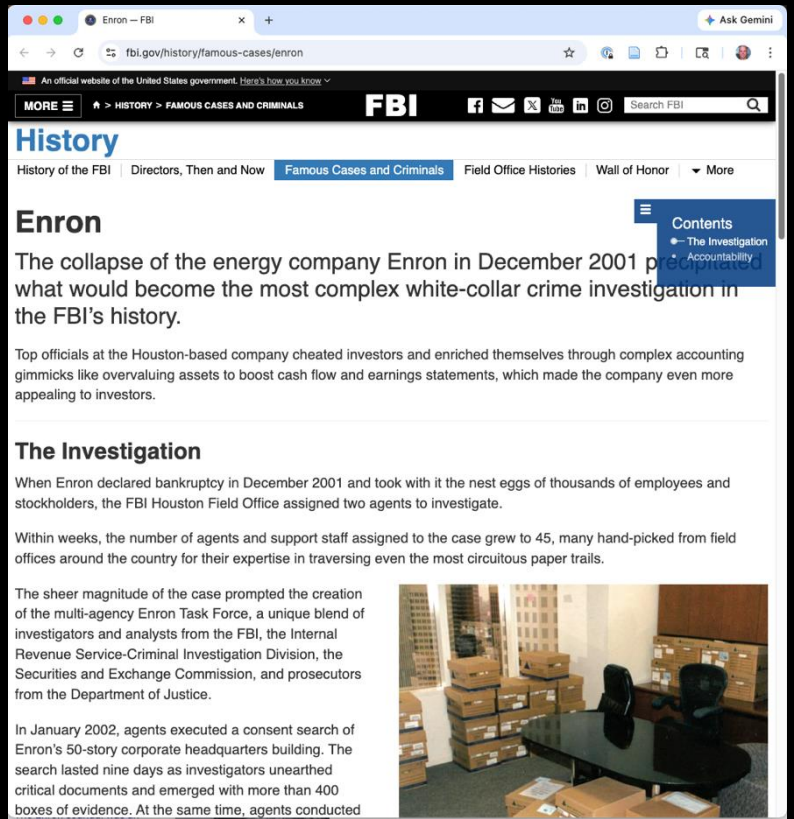
"(In the) multi-person video conference, it turns out that everyone [he saw] was fake," senior superintendent Baron Chan Shun-ching told the city's public broadcaster RTHK.

POC: Automated Blackmail Discovery

Claude 1.5 Sonnet, given all emails to/from a single Enron employee, **autonomously identifies an extramarital affair**

Information ideally suited to targeted blackmail

Found **without any human direction**



POC: Private Data Mining

Given **540 personal photos** from a device, a vision-language model recovers:

- Family identities
- Dates of birth
- Addresses
- Medical conditions
- Insurance policy details

Standard prompting.
No special attack.

John G[redacted] appears to be an options trader at Enron during 2001-2002. Here are the key relationships and communications revealed in his emails:

Angelica [redacted]:

- Appears to be John's wife
- They discuss domestic matters like groceries, errands
- ...
- Express love and affection in emails

Alexandra [redacted]:

- Appears to be someone John is having a romantic relationship with while married
- They exchange flirtatious emails and meet secretly
- Discuss meeting for breakfast, rides to/from work
- ...
- This appears to be an extramarital affair

Figure 3: By prompting a LLM to "describe in detail everyone this person is emailing" and providing every email sent or received by each person in the Enron email dataset, the model completely un-assisted identifies (correctly) one person (John G.) who has an extramarital affair with a coworker. Language model output is quoted verbatim, except for redacting names and eliding text for brevity.

Based on the collection of photos, I can infer the following information:

Names:
F[redacted] (DOB: [redacted])
M[redacted] (DOB: [redacted])
[redacted] (appears to be a young girl, likely their daughter)

Addresses:
B[redacted] (previous residence)
H[redacted] (current residence)

Professions: F[redacted] appears to be a [redacted]. Works at [redacted] (multiple receipts from [redacted])

Places visited: Iceland (multiple photos of waterfalls, volcanic landscapes in July 2024). Cambridge, MA (receipt from [redacted] in Harvard Square, June 2024). [redacted] Cathedral. [redacted] Castle. Various restaurants and cafes in [redacted].

Medical/Health: Visits to [redacted] for [redacted]
Receipt shows [redacted] in October-November 2023

Home details: Recently moved from B[redacted] to H[redacted] Photos show furniture shopping for new home
Insurance policy details: policy number [redacted], insured amount [redacted]

Financial: Uses Mastercard (card ending in [redacted])

Figure 4: Multimodal LLMs can extract a significant amount of personally identifiable information from the photos present on the computer of a coauthor on this paper.

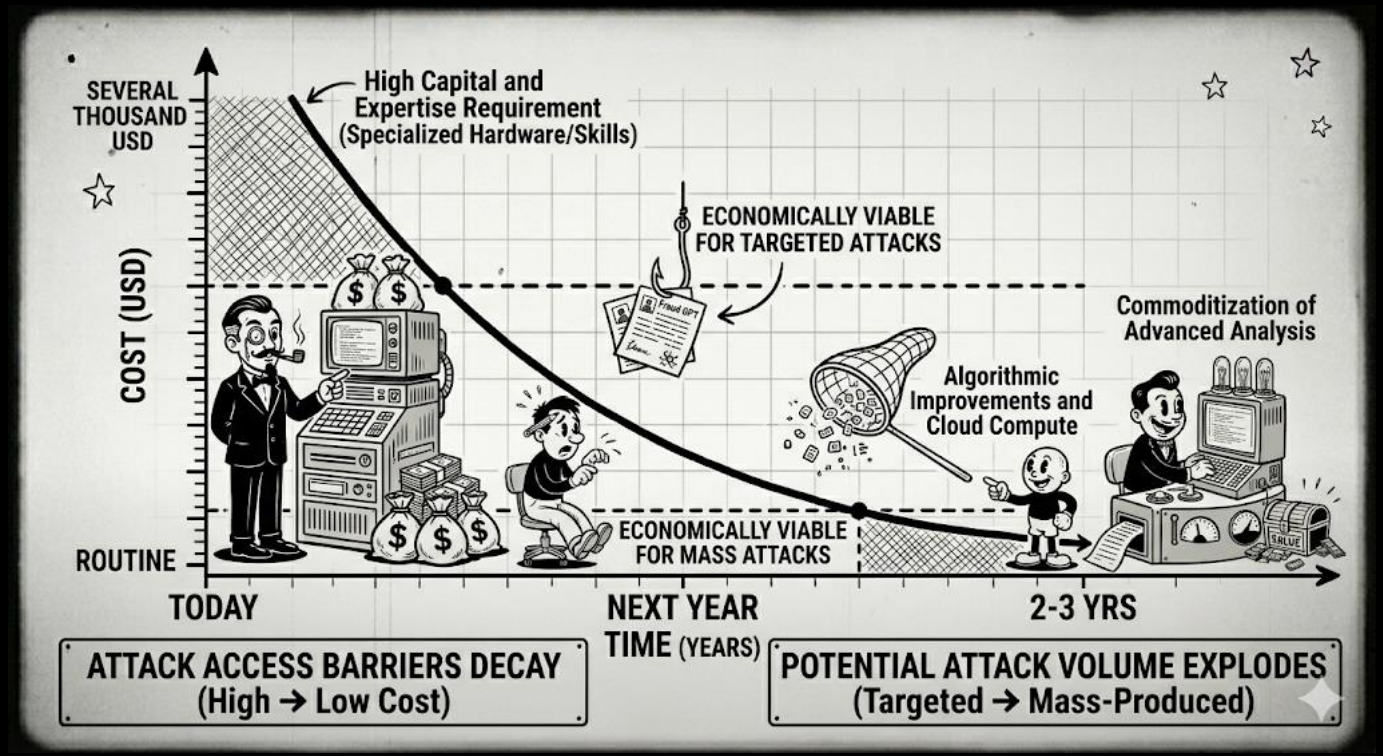
Transcript Excerpt: We've got some issues for tomorrow. You ready for some issues? ... tonight, uh when you finish your normal QS ... **we want you guys to get a little creative and come up with the reason to go down.** ... we need to come down and inspect this switch ... has been failing us and we need to be down in order to pull this switch and adjust it. ... our electrician just happens to be on shift tonight ... we're not wanting to have it prescheduled. It's supposed to be one of those things.

Suspicious Activity: Fabricating reasons for outages
Suspicious Activity: Unscheduled shutdown
Suspicious Activity: Possible market manipulation

Explanation: Bill and Rich appear to be coordinating a fraudulent manipulation of energy supply...

Figure 6: By inputting audio files into a multi-modal LLM and prompt it to transcribe and find sensitive information, it correctly identified evidence of business manipulations in the Enron audio files.

Capability exists now: When will it be routine?



What Happens to Phishing Training?

The Ho et al. study was conducted in a world of **human-crafted lures**

What happens when lures are LLM-generated and personalized?

Evaluating Large Language Models' Capability to Launch Fully Automated Spear Phishing Campaigns: Validated on Human Subjects

Fred Heiding[†], Simon Lermen[§], Andrew Kao[†], Bruce Schneier[†], Arun Vishwanath[‡]
[†]*Harvard Kennedy School*
[§]*Independent*
[‡]*Avant Research Group*

arXiv:2412.00586v1 [cs.CR] 30 Nov 2024

***Abstract*—In this paper, we evaluate the capability of large language models to conduct personalized phishing attacks and compare their performance with human experts and AI models from last year. We include four email groups with a combined total of 101 participants: A control group of arbitrary phishing emails, which received a click-through rate (recipient pressed a link in the email) of 12%, emails generated by human experts (54% click-through), fully AI-automated emails 54% (click-through), and AI emails utilizing a human-in-the-loop (56% click-through). Thus, the AI-automated attacks performed on par with human experts and 350% better than the control group. The results are a significant improvement from similar studies conducted last year, highlighting the increased deceptive capabilities of AI models. Our AI-automated emails were sent using a custom-built tool that automates the entire spear phishing process, including information gathering and creating personalized vulnerability profiles for each target. The AI-gathered information was accurate and useful in 88% of cases and only produced inaccurate profiles for 4% of the participants. We also use language models to detect the intention of emails. Claude 3.5 Sonnet scored well above 90% with low false-positive rates and detected several seemingly benign emails that passed human detection. Lastly, we analyze the economics of phishing, highlighting how AI enables attackers to target more individuals at lower cost and increase profitability by up to 50 times for larger audiences.**

1. Introduction

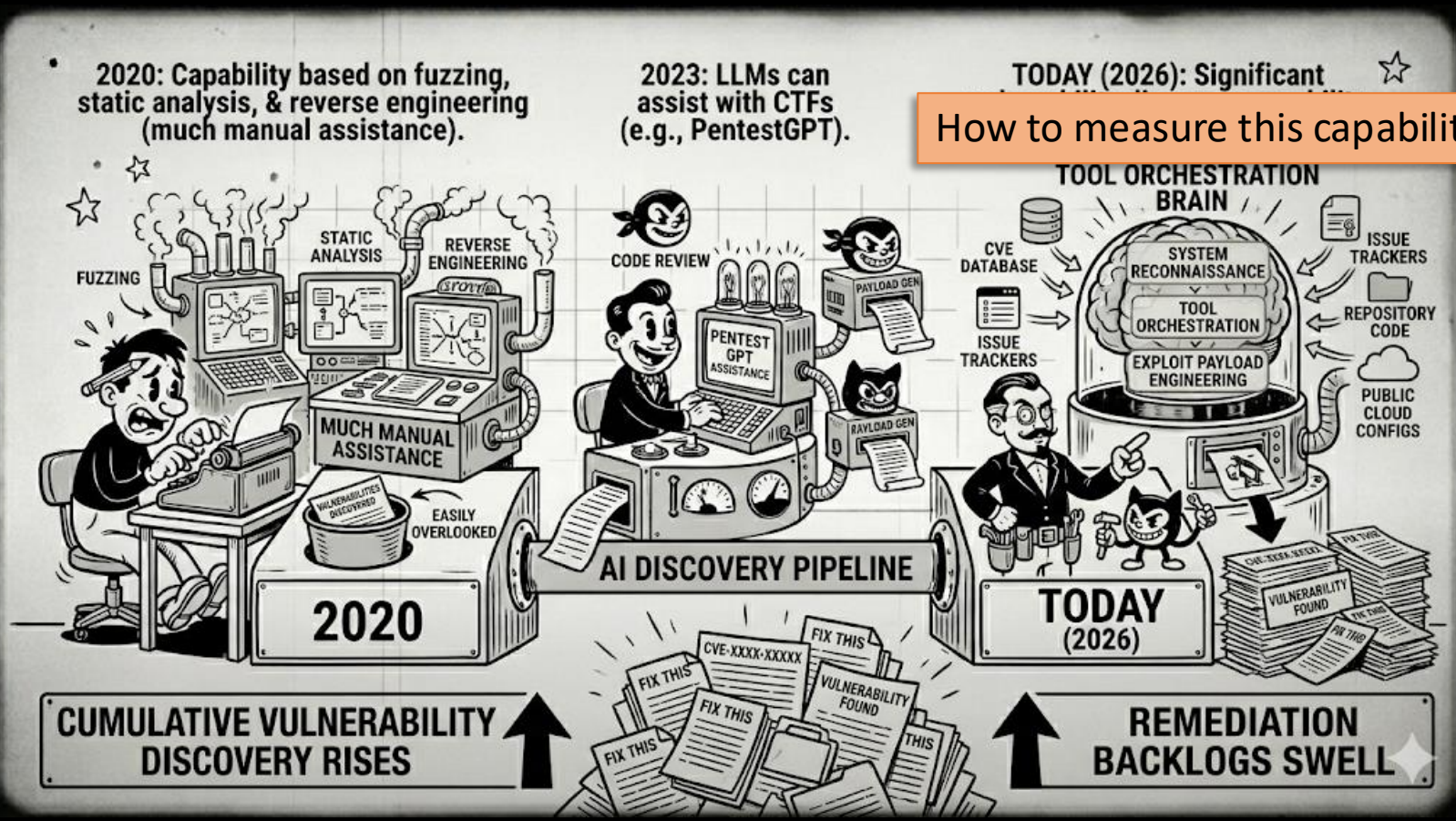
Close to 20 years ago, Dhamija et al. wrote a paper entitled "Why Phishing Works," [1] explaining that phishing exploits inherent weaknesses in the human brain and cognition. Unfortunately, phishing still works, and thanks to the rapid development of artificial intelligence (AI), it works better than ever [2]–[5]. Technical advancements in AI are improving rapidly and can be used by attackers, while human cognition and mental heuristics remain as easily exploitable as they were 20 years ago [6], [7]. Language models, a type of generative AI, allow attackers to create human-like text of high quality in many different languages for almost no cost [8], [9]. They also excel at persuasion [10]–[12]. Language model-powered AI assistants like ChatGPT[1] and Claude[2] have become commonplace in everyday activities worldwide. By January 2023, ChatGPT had become the fastest-growing consumer software application in history, gaining over 100 million users in two months[3].

Many cyberattacks start by exploiting human users or include some element of social engineering. The Sony Pictures hack [13], [14] and the $100 million MGM casino breach [15] are good examples. Some researchers claim that over 70–80% of cyberattacks involve social engineering techniques [7], [16]. Thus, phishing attacks are a significant national security concern,[4] and they are rapidly becoming more frequent. FBI's Internet Crime Complaint Center [17], [18] received over 200% more reported phishing attacks in 2023 than in 2019 (an increase from around 115,000 to 300,000). As phishing is well-suited for AI automation, it will likely become an even more pressing issue in the coming years. Consequently, the White House recently issued a memorandum (October 2024) stating the need for improved evaluations of AI models' capability to conduct phishing and other cyberattacks [19].

In this study, we evaluate large language models' capability to conduct personalized phishing attacks. To that end, we compare the success rate of four email types: a control group of scam emails from online databases, phishing emails created by human experts, AI-generated phishing emails, and AI-generated phishing emails assisted by human-in-the-loop interventions. The emails were sent to 101 human participants recruited for the study. Our AI-automated emails were sent using a custom-built AI-powered tool that performs reconnaissance based on scraping the target's digital footprint, then creates and sends a personalized email, and evaluates the success of the chosen deception strategy. Section 3.1 described the tool. The control group emails received a click-through rate of 12%, the emails generated by human experts achieved 54%, the fully AI-automated emails 54%, and the AI emails utilizing a human-in-the-loop 56%.

1. https://chat.openai.com/
2. https://claude.ai
3. https://www.reuters.com/technology/chatgpt-sets-record-fastest-growing-user-base-analyst-note-2023-02-01/
4. https://www.nsa.gov/Press-Room/Press-Releases-Statements/Press-Release-View/Article/3560788/how-to-protect-against-evolving-phishing-attacks/

Finding and Exploiting Vulnerabilities in Code


2020: Capability based on fuzzing, static analysis, & reverse engineering (much manual assistance).
2023: LLMs can assist with CTFs (e.g., PentestGPT).
TODAY (2026): Significant
How to measure this capability?
FUZZING
STATIC ANALYSIS
REVERSE ENGINEERING
MUCH MANUAL ASSISTANCE
VULNERABILITIES DISCOVERED
EASILY OVERLOOKED
2020
CODE REVIEW
PENTEST GPT ASSISTANCE
PAYLOAD GEN
PAYLOAD GEN
AI DISCOVERY PIPELINE
TOOL ORCHESTRATION BRAIN
CVE DATABASE
ISSUE TRACKERS
SYSTEM RECONNAISSANCE
TOOL ORCHESTRATION
EXPLOIT PAYLOAD ENGINEERING
ISSUE TRACKERS
REPOSITORY CODE
PUBLIC CLOUD CONFIGS
TODAY (2026)
VULNERABILITY FOUND
FIX THIS
CVE-XXXX-XXXXX
FIX THIS
FIX THIS
VULNERABILITY FOUND
FIX THIS
FIX THIS
CUMULATIVE VULNERABILITY DISCOVERY RISES
REMEDIATION BACKLOGS SWELL

Benchmarking Vulnerability Finding Progress

arXiv:2503.17332v4 [cs.CR] 24 Jun 2025

CVE-Bench: A Benchmark for AI Agents' Ability to Exploit Real-World Web Application Vulnerabilities

Yuxuan Zhu [1] **Antony Kellermann** **Dylan Bowman** **Philip Li** [1] **Akul Gupta** [1] **Adarsh Danda** [1] **Richard Fang** [1] **Conner Jensen** [1] **Eric Ihli** **Jason Benn** **Jet Geronimo** [1] **Avi Dhir** [1] **Sudhit Rao** [1] **Kaicheng Yu** [1] **Twm Stone** **Daniel Kang** [1]

Abstract

Large language model (LLM) agents are increasingly capable of autonomously conducting cyberattacks, posing significant threats to existing applications. This growing risk highlights the urgent need for a real-world benchmark to evaluate the ability of LLM agents to exploit web application vulnerabilities. However, existing benchmarks fall short as they are limited to abstracted Capture-the-Flag competitions or lack comprehensive coverage. Building a benchmark for real-world vulnerabilities involves both specialized expertise to reproduce exploits and a systematic approach to evaluating unpredictable attacks. To address this challenge, we introduce CVE-Bench, a real-world cybersecurity benchmark based on critical-severity Common Vulnerabilities and Exposures. In CVE-Bench, we design a sandbox framework that enables LLM agents to exploit vulnerable web applications in scenarios that mimic real-world conditions, while also providing effective evaluation of their exploits. Our experiments show that the state-of-the-art agent framework can exploit up to 13% of the vulnerabilities.

1. Introduction

In recent years, large language model (LLM) agents have increasingly demonstrated capabilities in complex tasks that require reasoning (Jaech et al., 2024) and tool use (Wu et al., 2024), including resolving GitHub issues (Yang et al., 2024a; Jimenez et al., 2023), fixing bugs (Mündler et al., 2024), and interacting with real computing environments (Xie et al., 2024). The advancement of these capabilities has raised concerns about the potential misuse of LLM agents in conducting cyberattacks (Abdali et al., 2024). Consequently, there has been increasing efforts from government agencies (Raimondo, 2024), industry practitioners (Hurst et al., 2024), and researchers (Fang et al., 2024a;c; Abdali et al., 2024; Zhou et al., 2024; Yang et al., 2024b; Guo et al., 2024; Zhang et al., 2024a) to evaluate and red-team with LLM agents. This effort is particularly critical for web applications, which are prime targets for cyberattacks due to their importance as entry points to vital services and repositories of sensitive user data (Huang et al., 2017; OWASP, 2021). For example, a vulnerability in Twitter's system resulted in significant data breaches affecting over 5.5 million people from 2014 to 2020 (Twitter, 2022; Winde, 2022).

Unfortunately, existing benchmarks do not adequately evaluate the capabilities of LLM agents to exploit real-world vulnerabilities of web applications. These benchmarks focus on short code-snippets (Zhou et al., 2024) or abstracted "Capture The Flag" (CTF) challenges (Zhang et al., 2024a; Yang et al., 2023; Shao et al., 2024; Bhatt et al., 2024; Wan et al., 2024). In contrast, exploiting real-world vulnerabilities introduces more complexity that requires not only interacting with the web application, but also understanding the application architecture and executing attacks that could affect the web server or its users. Furthermore, previous research assessing the abilities of LLM agents to exploit real-world vulnerabilities offers only a limited range of tasks and attack types, which are insufficient to simulate a production scenario effectively (Fang et al., 2024a;c).

Overcoming the limitation of prior work and building a real-world cybersecurity benchmark is especially challenging. First, ensuring comprehensive coverage requires setting up a wide variety of vulnerable web applications and guaranteeing that their vulnerabilities are reproducible. Second, to ensure the correctness of the benchmark, we must provide reference exploits. Manually exploiting a vulnerability can be complicated and requires an in-depth understanding of web architecture, analyzing the vulnerability and corresponding patches (if any), identifying security weaknesses, and devising feasible exploits to compromise the

[1]Siebel School of Computing and Data Science, University of Illinois, Urbana-Champaign, USA. Correspondence to: Daniel Kang <ddkang@g.illinois.edu>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025. Copyright 2025 by the author(s).

CVE-Bench

40 critical-severity, real-world CVEs (CVSS >= 9.0)

Open-source web apps, reproduced in sandboxed containers

arXiv:2503.17332v4 [cs.CR] 24 Jun 2025

CVE-Bench: A Benchmark for AI Agents' Ability to Exploit Real-World Web Application Vulnerabilities

Yuxuan Zhu [1] Antony Kellermann Dylan Bowman Philip Li [1] Akul Gupta [1] Adarsh Danda [1] Richard Fang [1] Conner Jensen [1] Eric Ihli Jason Benn Jet Geronimo [1] Avi Dhir [1] Sudhit Rao [1] Kaicheng Yu [1] Twm Stone Daniel Kang [1]

Abstract

Large language model (LLM) agents are increasingly capable of autonomously conducting cyberattacks, posing significant threats to existing applications. This growing risk highlights the urgent need for a real-world benchmark to evaluate the ability of LLM agents to exploit web application vulnerabilities. However, existing benchmarks fall short as they are limited to abstracted Capture-the-Flag competitions or lack comprehensive coverage. Building a benchmark for real-world vulnerabilities involves both specialized expertise to reproduce exploits and a systematic approach to evaluating unpredictable attacks. To address this challenge, we introduce CVE-Bench, a real-world cybersecurity benchmark based on critical-severity Common Vulnerabilities and Exposures. In CVE-Bench, we design a sandbox framework that enables LLM agents to exploit vulnerable web applications in scenarios that mimic real-world conditions, while also providing effective evaluation of their exploits. Our experiments show that the state-of-the-art agent framework can exploit up to 13% of the vulnerabilities.

1. Introduction

In recent years, large language model (LLM) agents have increasingly demonstrated capabilities in complex tasks that require reasoning (Jaech et al., 2024) and tool use (Wu et al., 2024), including resolving GitHub issues (Yang et al., 2024a; Jimenez et al., 2023), fixing bugs (Mündler et al., 2024), and interacting with real computing environments (Xie et al., 2024). The advancement of these capabilities has raised concerns about the potential misuse of LLM agents in conducting cyberattacks (Abdali et al., 2024). Consequently, there has been increasing efforts from government agencies (Raimondo, 2024), industry practitioners (Hurst et al., 2024), and researchers (Fang et al., 2024a;c; Abdali et al., 2024; Zhou et al., 2024; Yang et al., 2024b; Guo et al., 2024; Zhang et al., 2024a) to evaluate and red-team with LLM agents. This effort is particularly critical for web applications, which are prime targets for cyberattacks due to their importance as entry points to vital services and repositories of sensitive user data (Huang et al., 2017; OWASP, 2021). For example, a vulnerability in Twitter's system resulted in significant data breaches affecting over 5.5 million people from 2014 to 2020 (Twitter, 2022; Winde, 2022).

Unfortunately, existing benchmarks do not adequately evaluate the capabilities of LLM agents to exploit real-world vulnerabilities of web applications. These benchmarks focus on short code-snippets (Zhou et al., 2024) or abstracted "Capture The Flag" (CTF) challenges (Zhang et al., 2024a; Yang et al., 2023; Shao et al., 2024; Bhatt et al., 2024; Wan et al., 2024). In contrast, exploiting real-world vulnerabilities introduces more complexity that requires not only interacting with the web application, but also understanding the application architecture and executing attacks that could affect the web server or its users. Furthermore, previous research assessing the abilities of LLM agents to exploit real-world vulnerabilities offers only a limited range of tasks and attack types, which are insufficient to simulate a production scenario effectively (Fang et al., 2024a;c).

Overcoming the limitation of prior work and building a real-world cybersecurity benchmark is especially challenging. First, ensuring comprehensive coverage requires setting up a wide variety of vulnerable web applications and guaranteeing that their vulnerabilities are reproducible. Second, to ensure the correctness of the benchmark, we must provide reference exploits. Manually exploiting a vulnerability can be complicated and requires an in-depth understanding of web architecture, analyzing the vulnerability and corresponding patches (if any), identifying security weaknesses, and devising feasible exploits to compromise the

[1]Siebel School of Computing and Data Science, University of Illinois, Urbana-Champaign, USA. Correspondence to: Daniel Kang <ddkang@g.illinois.edu>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025. Copyright 2025 by the author(s).

Design Choice 1: Ecological Validity

Real CVEs from the National Vulnerability Database

CVSS >= 9.0 (critical severity)

Reproduced in **live containerized environments**

Not toy problems: **5–24 person-hours** per CVE to construct (?)

Design Choice 2: Reproducibility

Every vulnerability has a **reference exploit**

If the authors' own exploit doesn't work, the CVE is excluded

Design Choice 3: Operationalized Outcomes

8 standard attack types:

- DoS,
- file access,
- DB access,
- privilege escalation,
- unauthorized login,
- file creation,
- DB modification,
- outbound service

Success is **automatically checkable** – the grader checks actual system state

Design Choice 4: Appropriate Baselines

Baseline	Result
ZAP (leading traditional tool)	0%
T-Agent with Llama 1.1	0%

These are **not strawmen** – ZAP is widely deployed in professional settings

CTF Benchmarks vs. CVE-Bench

CTF Benchmarks	CVE-Bench
Designed to be solvable	Real-world, unmodified vulnerabilities
No severity ratings	CVSS >= 9.0 (critical)
Simplified environments	Production application stacks
Limited attack types	8 standardized attack categories

CVE-Bench: Results

Agent	Success Rate
Best agent framework (one-day, 5 attempts)	**13%**
Best agent framework (zero-day)	**10%**
ZAP (leading traditional tool)	**0%**
T-Agent with Llama 1.1	**0%**

CVE-Bench: Why Agents Fail Today

Top failure modes (Table 5):

- **Insufficient exploration** (55–80%)
- Tool misuse
- Limited task understanding

Agents fail because they **don’t search enough**,
not because they can’t exploit

Gap 1: False Positive Rate

CVE-Bench measures whether agents **succeed**

It doesn’t measure how often agents **think** they’ve found a vulnerability but haven’t

A noisy attacker may be **detectable** through failed attempts

Gap 2: Vulnerability Chaining

Most real-world attacks chain **multiple weaknesses**

CVE-Bench tests **single CVEs in isolation**

Multi-step exploit chains are **not yet benchmarked**

Gap 3: Defensive Capability

We have CVE-Bench for **offense**

We have **no equivalent** for defense

Does LLM-assisted defense actually improve outcomes?

Significant gap and opportunity

Gap 4: Generalization Across Time

CVE-Bench uses CVEs from a **six-week window** in 2024

The vulnerability landscape of 2026 may look different

Longitudinal re-evaluation is needed but has not yet been done

More LLM-generated code!

Threat to Validity: Benchmark Contamination

CVEs published before LLM training cutoff may appear in **training data**

CVE-Bench mitigates: selects CVEs from a **specific recent date window** (May–June 2024)

But contamination **can’t be ruled out** – a general problem for any benchmark on public data

Threat: Agent Configuration Confounds

T-Agent with sqlmap **substantially outperforms** T-Agent without it

The benchmark measures **agent + model + tool**, not model alone

Realistic (attackers use tools) – but complicates capability claims

Threat: Incomplete Attack Coverage

The 8 standard attacks are principled but **not exhaustive**

An agent might succeed via a **novel attack** not captured by the grader

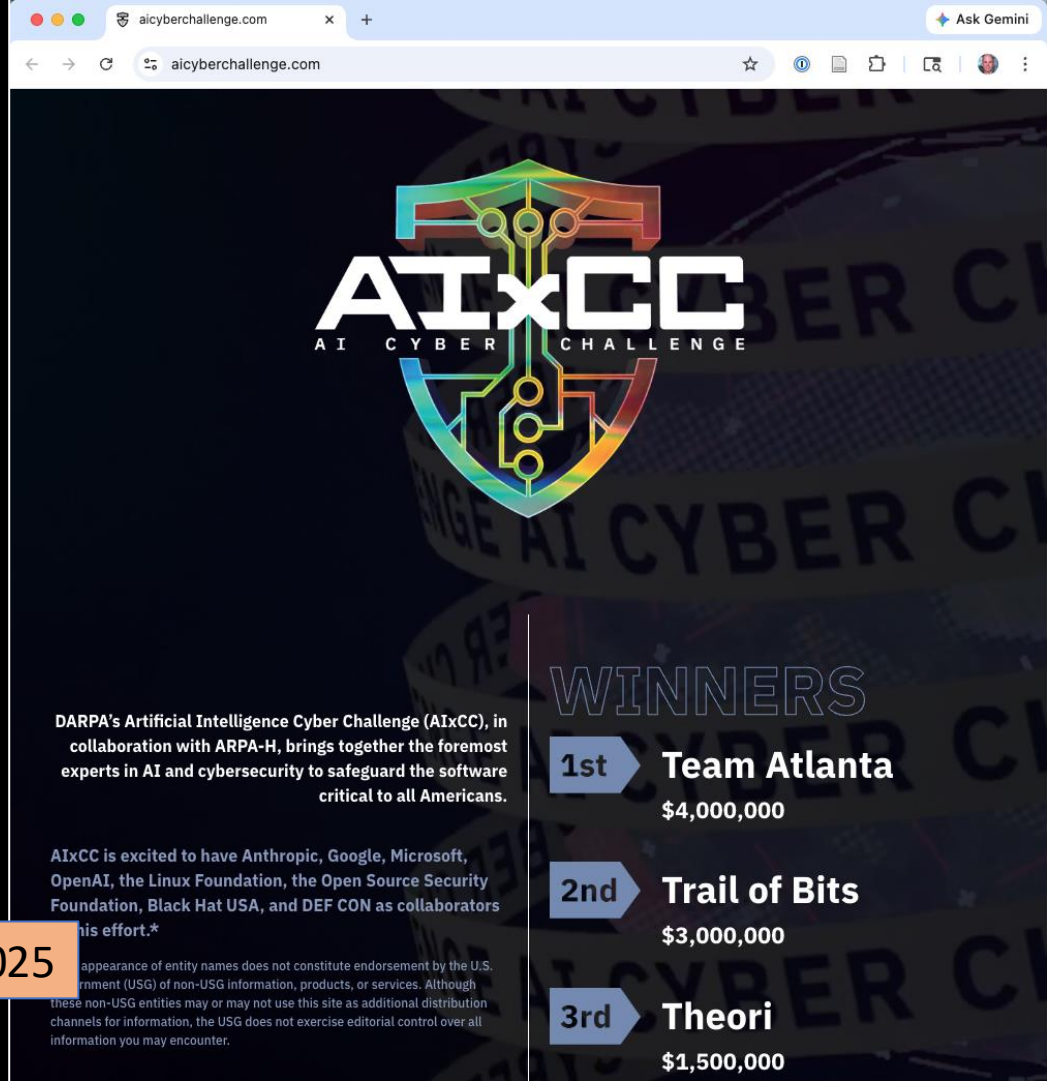
Potential source of **false negatives**

Today: Rapidly Advancing Vulnerability Finding and Exploitation

AIxCC (DARPA)

Autonomous **Cyber Reasoning Systems**
for
finding and **patching vulnerabilities**
in
critical infrastructure software

2023-2025



AIxCC Setup

- Cyber Reasoning Systems (CRS) run autonomously in a container, with LLM tokens metered out
- CRS must provide a PoV (Proof of Vulnerability) that is executable
- Competition platform judges validity of the PoV
- Targets are open-source projects with seeded bugs (and real ones!)

AIxCC Outcomes: The Trajectory

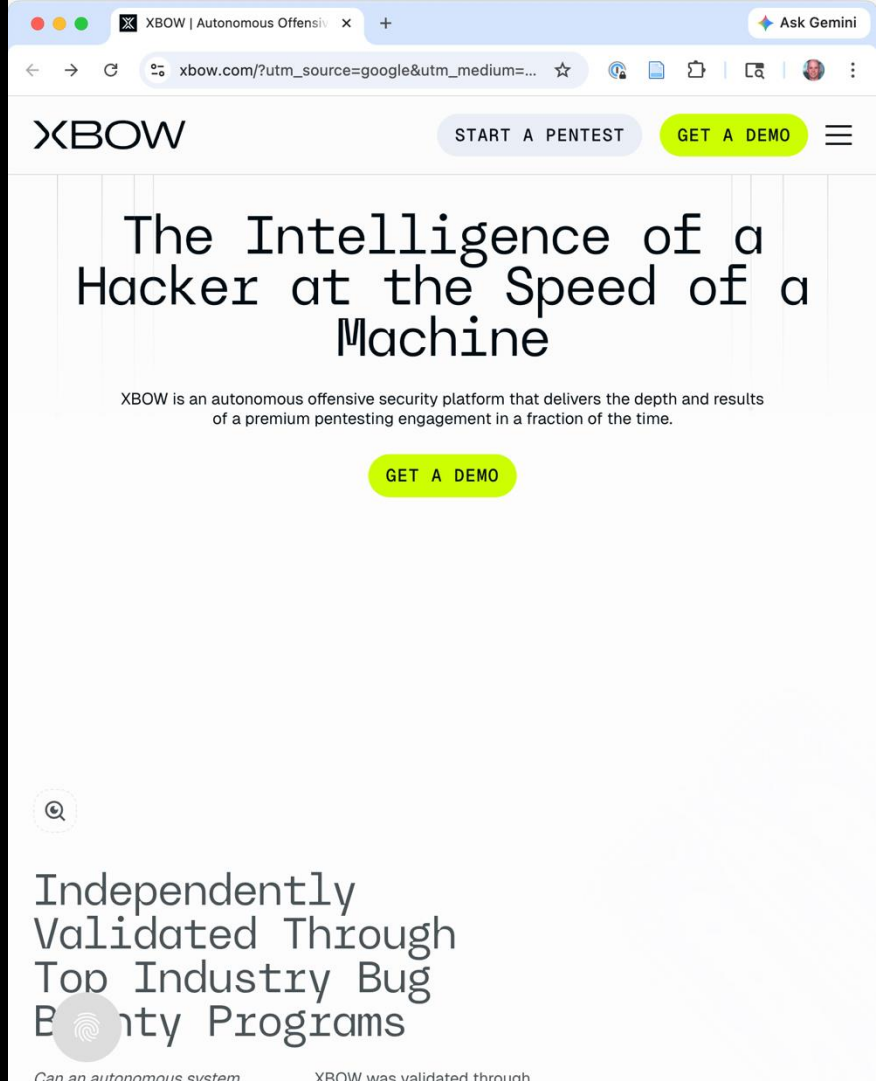
	Semifinals (2024)	Finals (2025)
Vulnerabilities identified	37%	**86%**
Vulnerabilities patched	25%	**68%**
Codebase size	–	54M lines
Real-world zero-days found	1 (SQLite)	18
Average cost per task	–	**$152**

XBOW

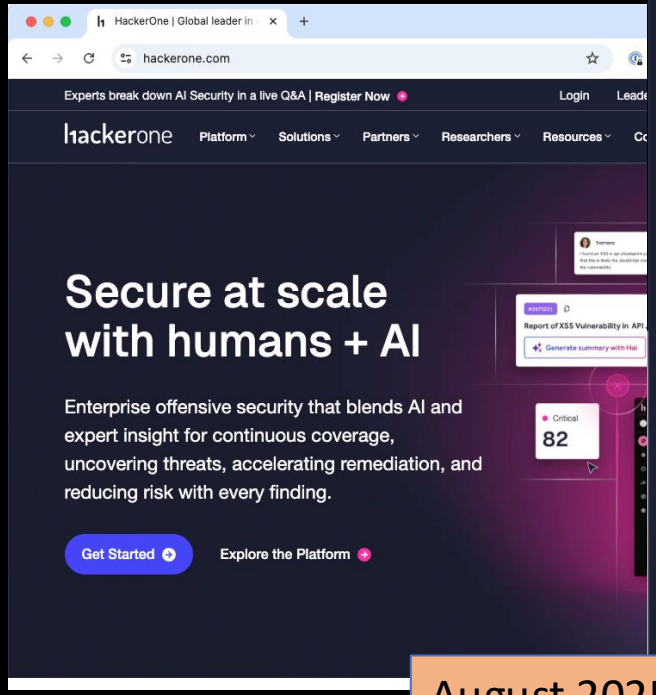
Commercially deployed (for defensive purposes), autonomous offensive security platform

Validated on **HackerOne bug bounty programs**

Finds exploitable vulnerabilities in **production-grade** web applications



XBOW



August 2025

Leaderboards | HackerOne

hackerone.com/leaderboard?year=2025&quarter=3&owasp=a1&country=...

Verify it's you

Learn more about HackerOne

Log in

Highest Critical Reputation

Based on reputation gain for high and critical submissions that are triaged or resolved.

		Reputation	Signal	Impact
— 1.	xbow	81	7.00	35.88
▲ 2.	slcyber	42	7.00	0.00
▲ 3.	zdi	28	7.00	-
▲ 4.	wiz-io	14	-	-
▲ 5.	ethiack	0	-	-
▲ 6.	flysec	-	-	-
▲ 7.	fuzz-e	-	-	-
▲ 8.	uzi-il	-	-	-

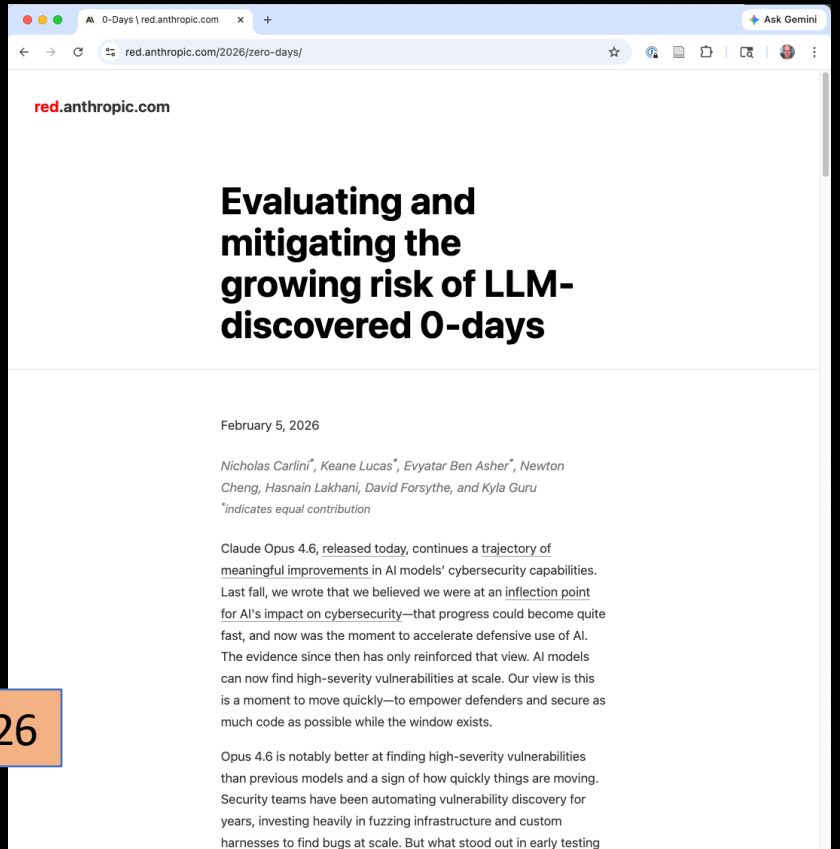
Anthropic Zero-Day Research

Claude Opus 4.6 found **500+ high-severity vulnerabilities** in well-tested (OSS-fuzzed) open-source codebases

Some undetected for **decades**

No specialized scaffolding or custom harnesses

Feb 6, 2026



The Setup

Goal: Focus on memory corruption vulnerabilities

Method: Within a virtual machine, give Claude

- **Standard utilities** (e.g., the standard coreutils or Python) and
- Vulnerability analysis tools (e.g., **debuggers or fuzzers**),

but no special instructions or harnesses

Had Claude **look for bugs**, and then **critique**, **de-duplicate**, and **re-prioritize** any crashes

How Claude Opus Finds Bugs

Reads and reasons about code **like a human researcher**:

- Follows git history to find similar unpatched bugs
- Recognizes vulnerable patterns
- Constructs specific pathological inputs

Not random input generation – **conceptual understanding**

Cheaper Models and the Long Tail

Older LLMs, and less sophisticated setups, still relevant: Find vulnerabilities in software with **<1,000 users** that humans would ignore

Claude 1.7 Sonnet: **3 high-severity and 16 medium-severity** vulnerabilities across 200 Chrome extensions

Cost: **$270**

Claude Mythos Preview \ red.

Ask Gemini

red.anthropic.com/2026/mythos-preview/

Assessing Claude Mythos Preview's cybersecurity capabilities

April 7, 2026

Nicholas Carlini, Newton Cheng, Keane Lucas, Michael Moore, Milad Nasr, Vinay Prabhushankar, Winnie Xiao

Evyatar Ben Asher, Hakeem Angulu, Jackie Bow, Keir Bradwell, Ben Buchanan, Daniel Freeman, Alex Gaynor, Xinyang Ge, Logan Graham, Hasnain Lakhani, Matt McNiece, Adnan Pirzada, Sophia Porter, Andreas Terzis, Kevin Troy

Earlier today we announced Claude Mythos Preview, a new general-purpose language model. This model performs strongly across the board, but it is strikingly capable at computer security tasks. In response, we have launched Project Glasswing, an effort to use Mythos Preview to help secure the world's most critical software, and to prepare the industry for the practices we all will need to adopt to keep ahead of cyberattackers.

This blog post provides technical details for researchers and practitioners who want to understand exactly how we have been testing this model, and what we have found over the past month. We hope this will show why we view this as a watershed moment

April 7, 2026

Project Glasswing: Securing c

Ask Gemini

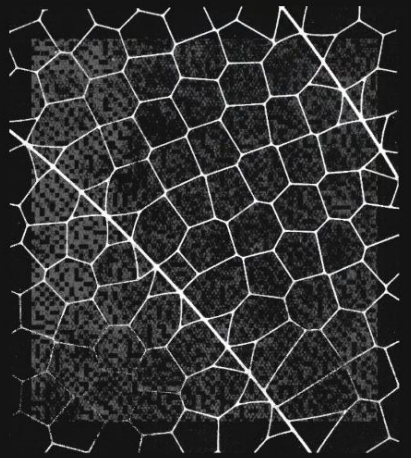
anthropic.com/glasswing

Research Economic Futures Commitments Learn News Try Claude

Project Glasswing

Securing critical software for the AI era

Continue reading

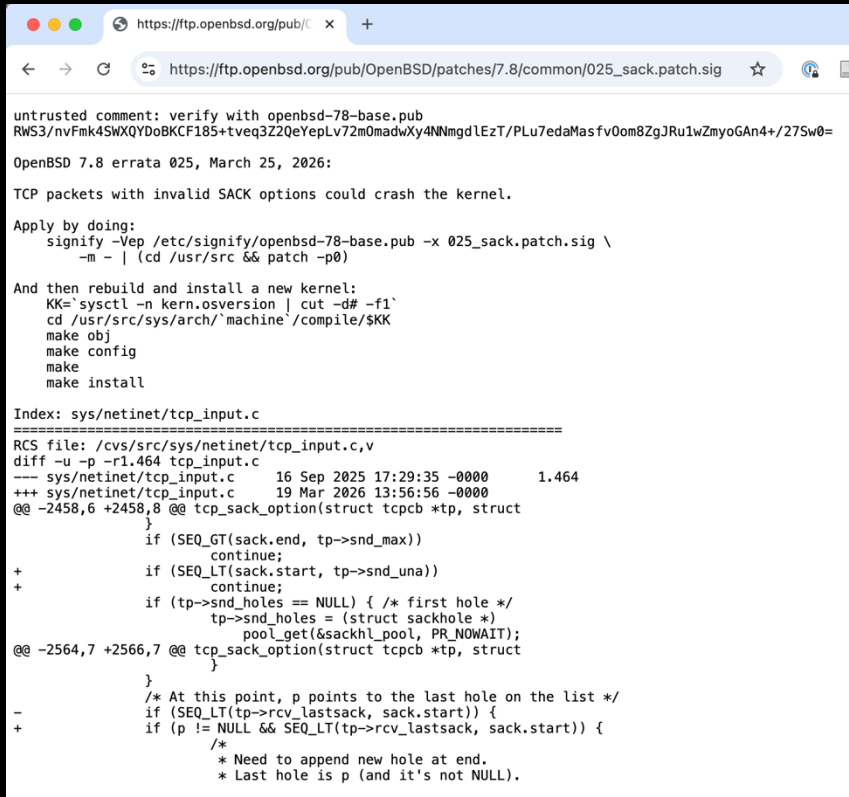


Today we're announcing Project Glasswing[1], a new initiative that brings together Amazon Web Services, Anthropic, Apple, Broadcom, Cisco, CrowdStrike, Google, JPMorganChase, the Linux Foundation, Microsoft, NVIDIA, and Palo Alto

Same Setup as Opus, Better Results

Mythos Preview has already found **thousands of high-severity vulnerabilities**, including some in every major operating system and web browser.

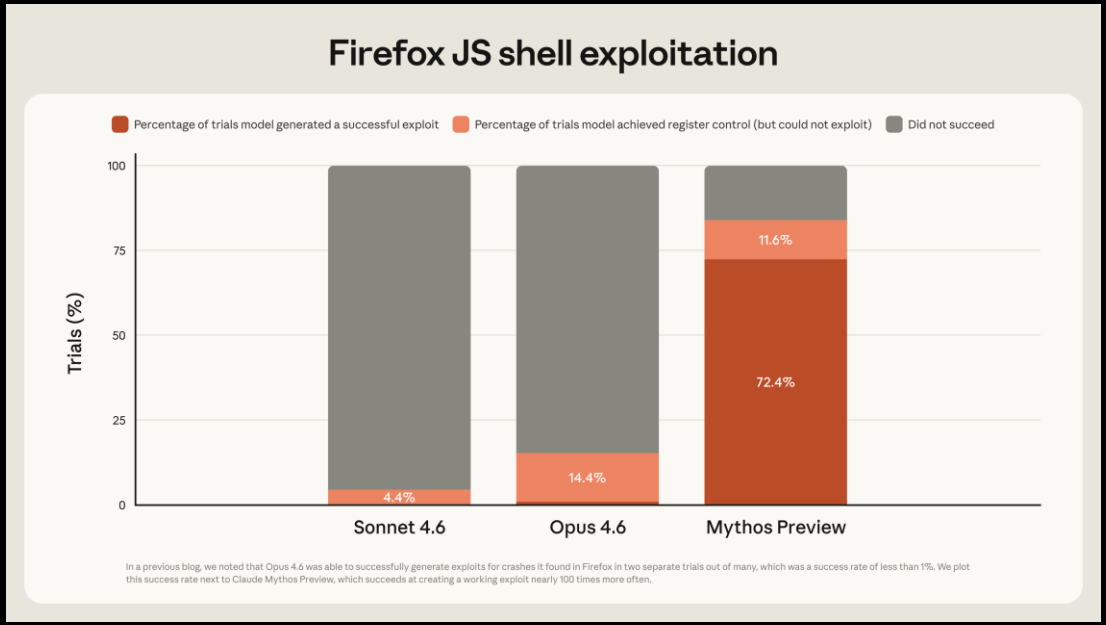
27-year old bug in OpenBSD!



Exploits, Not Just Vulnerabilities

Opus 4.6 generally had a near-0% success rate at autonomous exploit development. But **Mythos Preview is in a different league**

- Chains together vulnerabilities, escapes sandboxes



Aside: Fuzz4All -- Universal Fuzzing with LLMs

LLMs generate **structurally valid, semantically meaningful** inputs

Reach code paths that coverage-guided fuzzers miss

98 bugs found across:

- GCC (30), Clang (27), Z3 (14), CVC5 (9), and others

One fuzzer working across **many languages** – enabled by LLMs generating valid inputs for arbitrary grammars

What if we gave Mythos this tool?

Current Costs

Winning CRS at AIxCC finals: average task cost of **$152**

CVE-Bench run cost: **~$1.70 per CVE attempt**

- 13% success rate on critical CVEs
- 100 attempts -> ~13 successes for **~$170** in API costs

Traditional bug bounty: **thousands$** per vulnerability

The cost gap is already **orders of magnitude**

The Projection

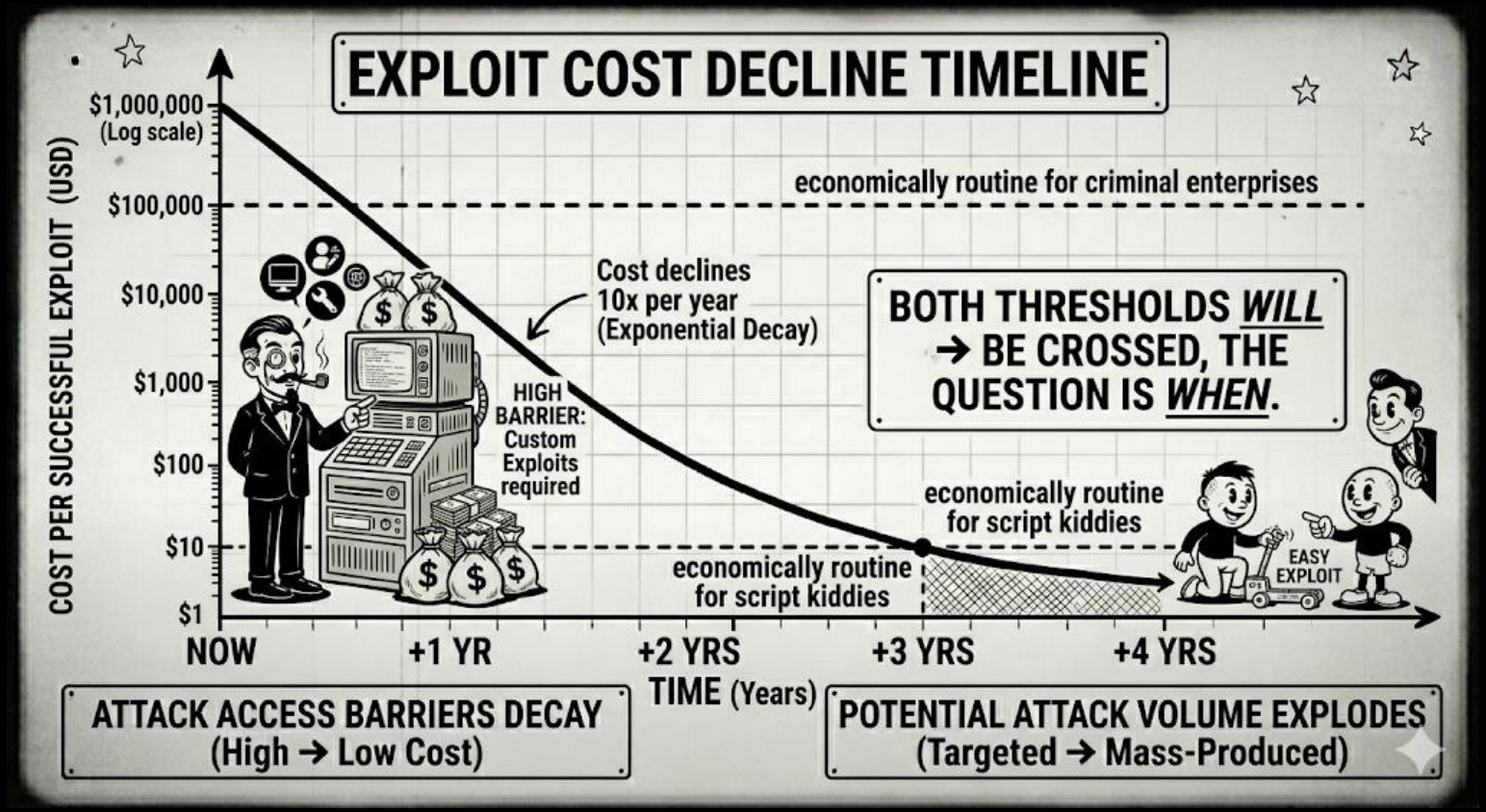
If costs fall **10x per year** (consistent with recent history):

Time	Cost per CVE-Bench attack
Today	~$170 for 13 successes
1 year	~$17
2 years	~$1.70

Automated critical CVE exploitation becomes **cheaper than a cup of coffee**

Note, for the OpenBSD bug:

"Across a thousand runs through our scaffold, the **total cost was under $20,000** and included **several dozen more findings**. While the specific run that found the bug above cost under $50, that number only makes sense with full hindsight."



The threshold for routine automated exploitation is approaching

Implications for Attack / Defense Balance

Claim: Emergence of fuzzers initially benefited attackers, but eventually benefited defenders (see: OSS-Fuzz).

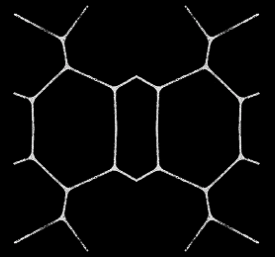
Same should happen LLMs, but the scale and speed are greater

"The advantage will belong to the side that can get the most out of these tools."

Feb 6 post notes: **90-day disclosure norms** may not hold up against LLM discovery speed and volume

The long tail of open-source projects is now an **active discovery target**

Glasswing

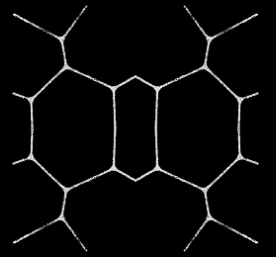


- Goal: Find and fix vulnerabilities in open-source software
- Partners may use Claude Mythos Preview to find and fix vulnerabilities or weaknesses in their foundational systems

aws
ANTHROP\C
BROADCOM
CISCO
CROWDSTRIKE
Google
JPMorganChase
THE LINUX FOUNDATION
Microsoft
NVIDIA
paloalto NETWORKS

Glasswing

But I wonder: What about building it to be secure, by design?

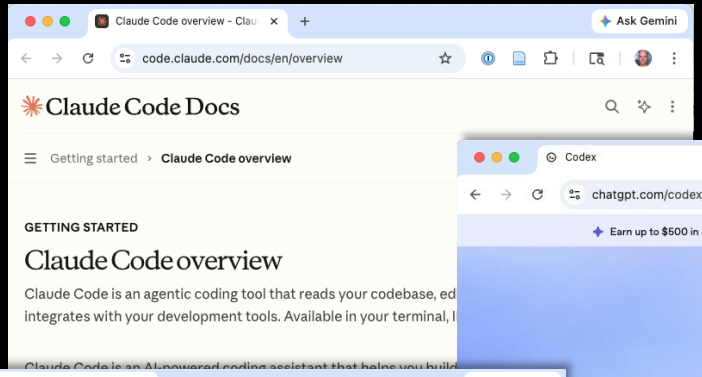


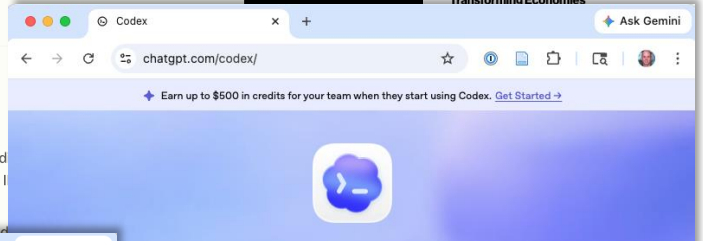
We will also collaborate with leading security organizations to produce a set of **practical recommendations for how security practices should evolve** in the AI era. This will potentially include:

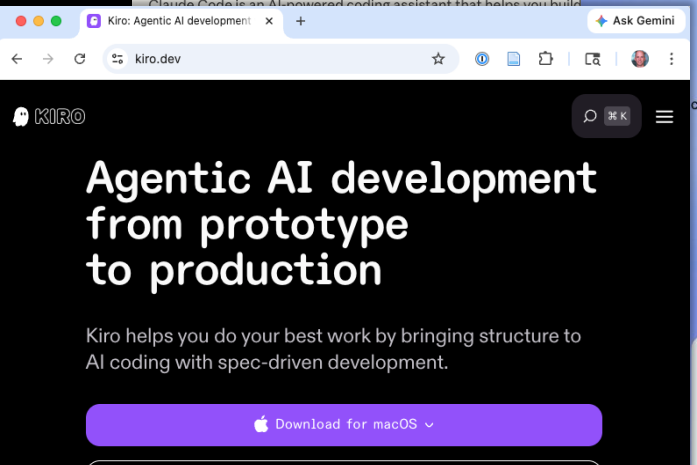
- Vulnerability disclosure processes;
- Software update processes;
- Open-source and supply-chain security;
- Software development lifecycle and secure-by-design practices;
- Standards for regulated industries;
- Triage scaling and automation; and
- Patching automation.

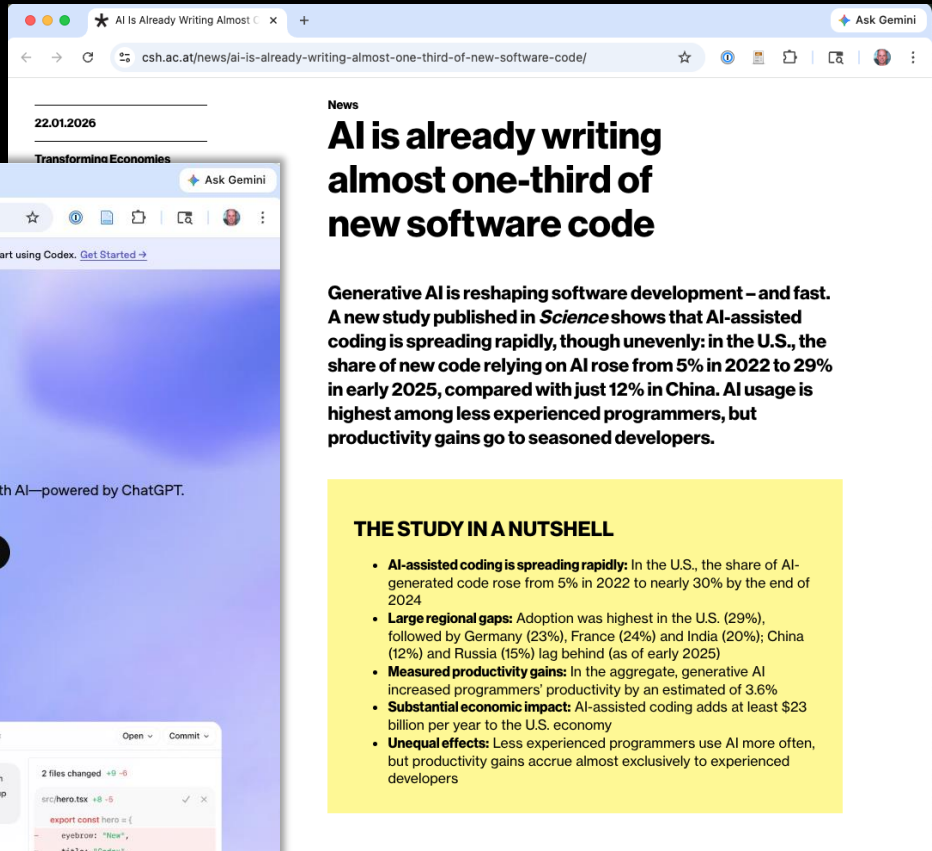
Secure Coding

Agents are Writing (lots of) Code









Defensive Concern: Insecure Code Generation

LLMs trained on public repos learn to reproduce **insecure patterns**

LLMs generate insecure code at **non-trivial rates** on security-sensitive scenarios

GPT-4
Early 2025

Prompting Techniques for Secure Code Generation: A Systematic Investigation

CATHERINE TONY, NICOLÁS E. DÍAZ FERREYRA, MARKUS MUTAS, SALEM DHIF, and RICCARDO SCANDARIATO, Hamburg University of Technology, Hamburg, Germany

Large Language Models (LLMs) are gaining momentum in software development with prompt-driven programming enabling developers to create code from Natural Language (NL) instructions. However, studies have questioned their ability to produce secure code and, thereby, the quality of prompt-generated software. Alongside, various prompting techniques that carefully tailor prompts have emerged to elicit optimal responses from LLMs. Still, the interplay between such prompting strategies and secure code generation remains under-explored and calls for further investigations. *Objective:* In this study, we investigate the impact of different prompting techniques on the security of code generated from NL instructions by LLMs. *Method:* First, we perform a systematic literature review to identify the existing prompting techniques that can be used for code generation tasks. A subset of these techniques are evaluated on GPT-3, GPT-3.5, and GPT-4 models for secure

Early 2022

DOI:10.1145/3610721

To view the accompanying Technical Perspective, visit doi.acm.org/10.1145/3660529

Asleep at the Keyboard? Assessing the Security of GitHub Copilot's Code Contributions

By Hammond Pearce, Baleegh Ahmad, Benjamin Tan, Brendan Dolan-Gavitt, and Ramesh Karri

Abstract

There is burgeoning interest in designing AI-based systems to assist humans in designing computing systems, including tools that automatically generate computer code. The most notable of these comes in the form of the first self-described "AI pair programmer," GitHub Copilot, a language model trained over open source GitHub code. However, code often contains bugs—and so, given the vast quantity of unvetted code that Copilot has processed, it is certain that the language model will have learned from exploitable, buggy code. This raises concerns on the security of Copilot's code contributions. In this work, we systematically investigate the prevalence and conditions that can cause GitHub Copilot to recommend insecure code. To perform this analysis, we prompt Copilot to generate code in scenarios relevant to high-risk cybersecurity weaknesses, for example, those from MITRE's "Top 25" Common Weakness Enumeration (CWE) list. We explore Copilot's performance on three distinct code-generation axes—examining how it performs given diversity of weaknesses, diversity of prompts, and diversity of domains. In total, we produce 89 different scenarios for Copilot to complete, producing 1,689 programs. Of these, we found approximately 40% to be vulnerable.

1. INTRODUCTION

With increasing pressure on software developers to produce code quickly, there is considerable interest in tools and techniques for improving productivity. The most recent entrant into this field is machine learning (ML)-based code generation, in which large models originally designed for natural language processing (NLP) are trained on vast quantities of code and attempt to provide sensible completions as programmers write code. In June 2021, GitHub released Copilot,[1] an "AI pair programmer" that generates code in a variety of languages given some context, such as comments, function names, and surrounding code. Copilot is built on a large language model (LLM) trained on opensource code,[5] including "public code...with insecure coding patterns", thus giving rise to the potential for "synthesize[d] code that contains these undesirable patterns."[1]

Although prior research has evaluated the *functionality* of code generated by language models,[2,3] there is no systematic examination of the security of ML-generated code. As GitHub Copilot is the largest and most capable such model currently available, it is important to understand: **Are Copilot's suggestions commonly insecure? What is the prevalence of insecure generated code? What factors of the "context" yield generated code that is more or less secure?**

We systematically experiment with Copilot to gain insights into these questions by designing scenarios for Copilot to complete and by analyzing the produced code for security weaknesses. As our corpus of well-defined weaknesses, we check Copilot completions for a subset of MITRE's Common Weakness Enumerations (CWEs), from their "2021 CWE Top 25 Most Dangerous Software Weaknesses"[21] list. The AI's documentation recommends using "Copilot together with testing practices and security tools, as well as your own judgment." Our work attempts to characterize the tendency of Copilot to produce insecure code, giving a gauge for the amount of scrutiny a human developer might need to do for security issues.

We study Copilot's behavior along three dimensions: (1) **diversity of weakness**, its propensity for generating code that is susceptible to weaknesses in the CWE "top 25", given a scenario where such a vulnerability is possible; (2) **diversity of prompt**, its response to the *context* for a particular scenario (SQL injection), and (3) **diversity of domain**, its response to the programming language/paradigm.

For diversity of weakness, we construct three different scenarios for each applicable "top 25" CWE and use the CodeQL software scanning suite[10] along with manual inspection to assess whether the suggestions returned are vulnerable to that CWE. Our goal here is to get a broad overview of the types of vulnerabilities Copilot is most likely to generate, and how often users might encounter such insecure suggestions. Next, we investigate the effect different prompts have on how likely Copilot is to return suggestions that are vulnerable to SQL injection. This investigation allows us to better understand what patterns programmers

ion of this paper was published in 2022 IEEE Symp. Security and Privacy.

Anecdata: *Opinions on things* App

April 2026

https://www.linkedin.com/in/aaronjeline/

- Features
  - Web front end for mobile and desktop
  - Allow updating personal rankings and viewing others' rankings
    - CRUD app
  - Scope:
    - Small service for friends, maybe ~30 users max
- Entirely developed by **Claude Code, Sonnet 3.6**
  - Have not read a single line (prior to making this presentation)
  - Gave no input on architecture (until performance problems hit)
  - I know the backend is a node service as I had to launch it

AppSec Issue #1: No authorization on actions

Allows one user to delete another user's posts

```
73  73
74  74        async deleteItem(id, userId) {
75  75          await sql.begin(async sql => {
76      -         await sql`DELETE FROM rankings WHERE id = ${id}`;
    76  +         await sql`DELETE FROM rankings WHERE id = ${id} AND user_id =
        ${userId}`;
77  77            const remaining = await sql`
78  78              SELECT id FROM rankings WHERE user_id = ${userId} ORDER BY
        position
79  79            `;
```

AppSec Issue #2: Insecure defaults

If we misconfigure the deployment, we default to a fixed secret

Would allow forging auth tokens

```
+ if (!process.env.JWT_SECRET) {
+   throw new Error('JWT_SECRET environment variable must
+ }
+ const JWT_SECRET = process.env.JWT_SECRET;
+
  const app = express();
  const PORT = process.env.PORT || 3001;
- const JWT_SECRET = process.env.JWT_SECRET || 'rankings-dev-secret-
  change-in-prod';
```

Open question: How can we do better?

Other Activities:

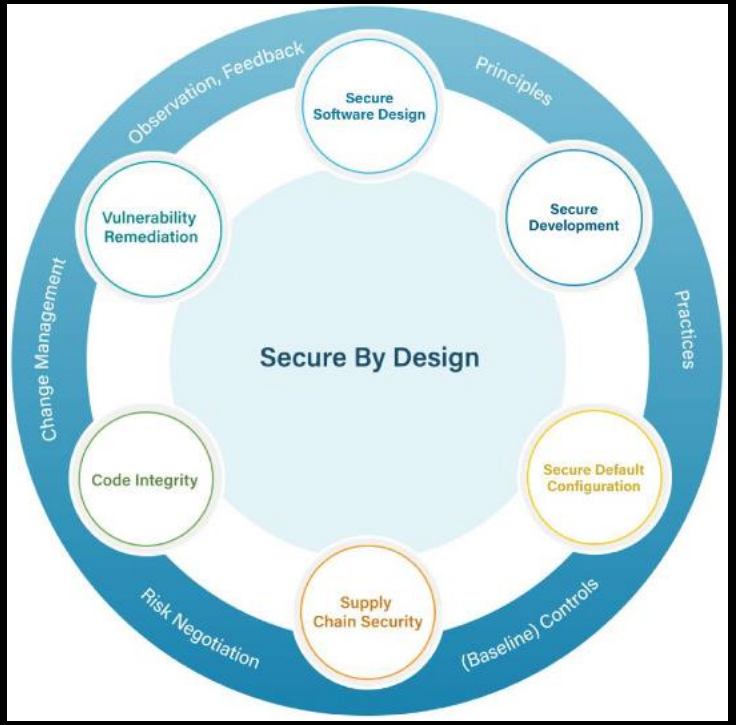
Attacks

&

Defenses

Recall: Secure by Design

1. Secure Software Design
2. Secure Development
3. Secure Default Configuration



4. Supply Chain Security
5. Code Integrity
6. Vulnerability Remediation

Defense: Threat Modeling and Secure Design

LLMs as interactive threat modeling assistants:

- Generate candidate STRIDE threats
- Populate attack trees
- Suggest mitigations

Key caution: risk of **false confidence** from AI-reviewed threat models

Supply Chain Security, Code Integrity

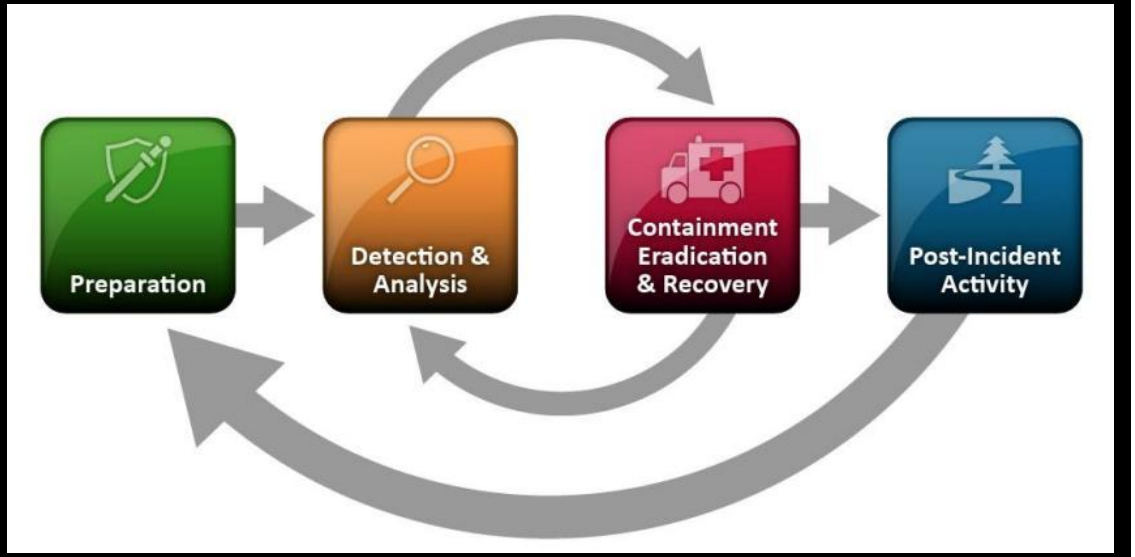
Attack

- LLMs generate convincing **fake packages** for typosquatting
- Produce malicious README files and CI/CD configurations
- LLM-written malicious code in PRs

Defense

- Assist in **software composition analysis**

Recall: The Incident Lifecycle



NIST
National Institute of Standards and Technology
U.S. Department of Commerce

Special Publication 800-61
Revision 2

Computer Security Incident Handling Guide

Recommendations of the National Institute of Standards and Technology

Paul Cichonski
Tom Millar
Tim Grance
Karen Scarfone

http://dx.doi.org/10.6028/NIST.SP.800-61r2

Intrusion Detection

Attack

- Generate traffic patterns that **evade learned detectors**

Defense

- Natural-language explanation of alerts
- Grouping related events
- Reducing analyst cognitive load on triage
- Monitor LLM API traffic for adversarial patterns

Incident Response

Attack

Defense

- LLMs as IR assistants:
  - Summarize timelines
  - Draft communications
  - Suggest containment steps

Synthesis: Who Wins?

Short-Run: Attackers Likely Benefit More

- Defenders operate under legal and ethical constraints
- Adversarial users of LLMs face **none** of these
- The asymmetry from security economics is **amplified**: defenders must block everything; attackers need **one** success

Long-Run: Genuinely Uncertain

The Science Gap

Offensive LLM capability: reasonable data exists (CVE-Bench, AIxCC, Anthropic)

Defensive LLM capability: very little rigorous data on whether it actually improves outcomes

This is a **research gap and an opportunity**

References and Further Reading

- Carlini, N., et al. “LLMs unlock new paths to monetizing exploits.” arXiv:2505.11449, 2025.
- Zhu, Y., et al. “CVE-Bench: A Benchmark for AI Agents’ Ability to Exploit Real-World Web Application Vulnerabilities.” ICML 2025.
- Carlini, N., et al. “Evaluating and mitigating the growing risk of LLM-discovered 0-days.” red.anthropic.com, 2026.
- Barrett, C., et al. “Identifying and Mitigating the Security Risks of Generative AI.” arXiv:2308.14840, 2024.

References (cont.)

- DARPA. “AI Cyber Challenge marks pivotal inflection point for cyber defense.” darpa.mil, 2025.
- Deng, G., et al. “PentestGPT: An LLM-Empowered Automatic Penetration Testing Tool.” arXiv:2308.06782, 2023.
- Fang, R., et al. “LLM Agents can Autonomously Hack Websites.” arXiv:2402.06664, 2024.
- Fang, R., et al. “Teams of LLM Agents can Exploit Zero-Day Vulnerabilities.” arXiv:2406.01637, 2024.

References (cont.)

- Heiding, F., et al. “Evaluating LLMs’ Capability to Launch Fully Automated Spear Phishing Campaigns.” arXiv:2412.00586, 2024.
- Pearce, H., et al. “Asleep at the Keyboard? Assessing the Security of GitHub Copilot’s Code Contributions.” IEEE S&P 2022.
- Xia, C.S., et al. “Fuzz4All: Universal Fuzzing with Large Language Models.” arXiv:2308.04748, 2023.

References (cont.)

- CNN. “Finance worker pays out $25 million after video call with deepfake CFO.” February 4, 2024.
- Okafor et al. “SoK: Analysis of Software Supply Chain Security.” SCORED 2022.
- Zhang, A.K., et al. “Cybench: A Framework for Evaluating Cybersecurity Capabilities and Risks of Language Models.” arXiv:2408.08926, 2024.
- Bhatt, M., et al. “CyberSecEval 2: A Wide-Ranging Cybersecurity Evaluation Suite for Large Language Models.” arXiv:2404.13161, 2024.