Measuring Safety in Autonomous Penetration Testing

Ethan Ho
Department of Computer Science
Stanford University
Stanford, CA 94305
ethanho@stanford.edu

Abstract

Existing cybersecurity benchmarks often measure whether language-model agents can solve technical tasks, but professional penetration testing also requires safe operation in business-critical environments. This project presents a pilot benchmark for measuring *safe cybersecurity capability*: making useful security progress while preserving availability and managing operational risk. We evaluate four models on two sandboxed scenarios, a SaaS web application and a simulated water-treatment industrial-control system, using a Kali-based Codex scaffold with a fixed time limit. Finding quality is scored by matching agent reports against severity-weighted gold findings. Safety is measured separately with two deterministic dimensions: availability and risk management, where risk management captures scenario-specific harms such as sensitive-data exposure, credential-sensitive views, and unsafe ICS state changes. The pilot run shows that models which find more vulnerabilities can also create larger operational safety costs. A Generalizability Theory analysis suggests meaningful model-level variation, but also a substantial model-by-scenario interaction, so the results should not be read as a definitive leaderboard. The main contribution is a measurement framework and pilot analysis showing how safe cybersecurity capability can be separated into finding quality, availability, and risk management, and why a larger scenario suite is needed for stronger claims.

1 Introduction

Current cybersecurity benchmarks often measure whether language-model agents can solve technical tasks, such as finding vulnerabilities or completing CTF-style challenges. Real penetration testers, however, often interact with business-critical systems where disruption can cause financial, safety, or public-health harms. Penetration testing therefore requires more than technical competence: the tester must stay within scope, avoid service disruption, minimize exposure of sensitive data, and escalate unexpected risks. This project studies *safe cybersecurity capability*: the ability of AI agents to make useful security progress while respecting operational safety constraints.

This study aims to investigate whether frontier language-model agents can make useful progress in realistic penetration-testing scenarios while preserving availability and managing operational risk, and how this skill can be measured from an AI measurement science point of view. We evaluate this question with a pilot benchmark containing controlled, sandboxed scenarios with explicit rules of engagement. Each scenario includes a target objective and realistic opportunities for both useful progress and unsafe behavior, such as disruptive testing, unnecessary exposure of sensitive data, credential misuse, or unsafe changes to an industrial-control system. Four models are evaluated, and their action transcripts are scored for both finding quality and safety.

Submitted for Stanford CS 321M.

2 Background

This project connects to AI measurement science through construct validity, consequential validity, and reliability. Existing cybersecurity benchmarks may validly measure vulnerability discovery or exploit completion, but those constructs are not identical to safe cybersecurity capability. This distinction matters because an agent can be technically successful while still causing confidentiality, integrity, availability, or client-trust harms. A benchmark that rewards only technical success may therefore have weak consequential validity if high scores encourage unsafe deployment. To address this measurement gap, we separate capability from safety by scoring finding quality alongside two deterministic safety dimensions: availability and risk management. We then use Generalizability Theory to estimate how much observed score variance comes from the model, the scenario, and their interaction.

Prior work has made substantial progress toward realistic cyber-agent evaluation. Cybench evaluates agents on professional-level CTF tasks with command execution and feedback [Zhang et al., 2024], while CVE-Bench and CyberGym use sandboxed vulnerabilities and real software vulnerability-reproduction tasks [Zhu et al., 2025, Wang et al., 2025]. CyberSecEval frames cybersecurity evaluation as both capability and risk measurement across secure-code, compliance, and offensive-cyber tasks [Wan et al., 2024]. Although there has been progress toward more realistic cybersecurity evaluations, these evaluations primarily emphasize whether agents complete technical objectives, not whether they behave like careful professional testers when progress conflicts with safety.

Professional security-testing guidance motivates treating safety as a first-class measurement outcome. NIST SP 800-115 emphasizes planning, authorization, rules of engagement, findings analysis, and mitigation [Scarfone et al., 2008]; the OWASP Web Security Testing Guide similarly situates web testing inside a controlled methodology and pre-engagement process [OWASP Foundation, 2024]. Industrial-control guidance makes the issue sharper: NIST SP 800-82 describes ICS, SCADA, DCS, and PLC environments as having distinctive reliability, performance, and safety requirements [Stouffer et al., 2015]. In these settings, actions that look acceptable in a toy exploit benchmark can be harmful if they disrupt a controller, change process variables, or reduce system availability.

The closest related study is ARTEMIS, which evaluates AI agents and human professionals in a live enterprise penetration-test environment and discusses operational risks such as confidentiality, integrity, availability, destructive actions, scope limits, and real-time monitoring [Lin et al., 2025]. My work is complementary: rather than asking only whether agents find valid vulnerabilities, this benchmark asks whether they can produce a useful report while preserving the operational constraints that make a penetration test professionally acceptable. The measurement design also follows broader language-model evaluation work such as HELM, which argues for transparent multi-metric evaluation rather than reducing model quality to a single accuracy number [Liang et al., 2022]. Because reported findings are open-ended, we use LLM-assisted semantic matching for finding quality, while keeping safety scoring deterministic and scenario-based [Zheng et al., 2023]. Finally, the reliability analysis draws on Generalizability Theory, which models measurement error across facets rather than treating it as one residual term [Cronbach et al., 1972, Shavelson and Webb, 1991, Brennan, 2001].

3 Methods

We built a pilot benchmark for evaluating safe cybersecurity capability in realistic penetration-testing workflows. The central construct is *safe cybersecurity capability*: the ability to make useful security progress while staying in scope, preserving service availability, minimizing sensitive-data exposure, handling credentials appropriately, and escalating when necessary.

Experimental setup. We evaluate models from OpenAI, Anthropic, Google, and Qwen. This mix allows us to compare safe cybersecurity capability across models with different deployment assumptions and guardrail profiles. All models are accessed through OpenRouter and are changable in a configuration file, so the experiment can be reproduced or extended without changing benchmark code.

Each scenario is a sandboxed Docker environment. Agents operate from an isolated Kali Linux container using Codex CLI for command execution and interaction with the target. Each model receives the same objective, environmental information, and rules of engagement. The prompt gives

an in-scope host and states that all services on that host are in scope, but it does not reveal target ports. It also asks for a final report with proof of exploitation for confirmed findings, including the exact command or request, concise output evidence, affected service, impact, and remediation. Agents are configured to run for thirty minutes per scenario. Commands, tool calls, observations, model messages, availability checks, and scenario checks are logged as timestamped transcripts.

For a batch experiment, the runner executes the full model-by-scenario matrix. Each cell receives its own Docker Compose project, Kali agent container, target container, transcript path, and timestamped run directory, so unsafe behavior in one run cannot contaminate another.

Scenarios. We construct 2 controlled penetration-test scenarios. The first is a SaaS-style business website: a PHP web application with login functionality, a backend database, and sensitive user information. It includes a SQL injection path for exploitation, but also unsafe opportunities such as excessive login attempts that may make the service unavailable or unnecessary exposure of PII.

The second is a simulated industrial-control system for a water-treatment plant. It includes SCADA and Modbus components and safety-relevant process variables such as chemical levels, pump state, tank level, controller mode, and actuator settings. Unsafe actions can degrade the simulated plant state; safer behavior requires cautious inspection, limited interaction, and escalation. All scenarios are sandboxed and do not interact with real external systems.

Agent workflow. The agent loop uses the OpenAI Codex scaffold. For each model-scenario cell, the harness starts Codex inside the Kali container with the scenario prompt and model ID. The harness observes Codex output, probes target availability, and enforces the time limit plus a shutdown grace period. Before shutdown, Codex is asked to produce a final report so even incomplete runs can be evaluated.

Evaluation metrics. We evaluate each run using two types of metrics.

First, we produce a score for *finding quality*. Each scenario has a gold report listing vulnerabilities present in the environment. After a run, LLM matchers semantically compare the agent's reported findings to the gold findings, producing matched findings, missed findings, reported findings, extra reported findings, finding recall, and a precision proxy. We assign each LLM a performance score based on how many gold findings were matched to LLM findings, as well as a weighted performance score using the same gold reports, but weighting each finding by its severity. Overall, this metric measures *capability*: a model should not be rated highly merely for avoiding risk while finding nothing.

Second, we produce scores for *operational safety*. Operational safety has two dimensions: *availability* and *risk management*.

Availability checks aim to objectively capture whether the agent causes operational disruption while testing. Every 20 seconds, an automated checker tests whether the target service remains functional from the perspective of a normal business user. For the industrial-control scenario, this will also test whether the chemicals in the water treatment plant are within safe levels. The availability score is

$$\text{Availability} = \frac{\#\text{ functionality checks passed}}{\#\text{ total functionality checks}}.$$

Risk-management checks encode concrete failure conditions for each environment, which are created to demonstrate business and operation critical failure modes that could happen during a penetration test. In the business-website scenario, risk management covers PII leakage into agent transcripts and credential-sensitive endpoint views. In the industrial-control scenario, risk management covers unsafe writes and other unsafe state changes. These events are converted into a weighted risk load L, and the risk-management score is

$$\text{Risk management} = \frac{1}{1+L}.$$

This continuous score avoids clipping all high-risk runs to zero while still penalizing larger exposures more heavily. When a single aggregate operational-safety score is needed, the risk-management score is applied as a multiplier on the availability score. Overall, these metrics measure *safety*: a model should not be rated highly merely for reporting better findings while being reckless in a production environment.

Connection to course material. This project applies course concepts from validity, reliability, and Generalizability Theory. The benchmark is motivated by a construct-validity concern: existing cybersecurity benchmarks may measure vulnerability discovery or exploit completion, but not safe cybersecurity capability under professional penetration-test constraints. Furthermore, the benchmark is motivated by a consequential validity concern, as high performance on penetration testing benchmarks may not neccesarily translate to usability in the real world with production systems.

The main quantitative analysis uses Generalizability Theory to estimate sources of measurement variance. We treat the evaluated model and scenario as facets of the measurement design. In the language of course measurement models, the observed score is not a direct estimate of a model's true ability; it is produced by a particular scenario. A simplified variance breakdown is

$$Y_{ms} = \mu + M_m + S_s + (MS)_{ms},$$

where Y_{ms} is the operational-safety score for model m on scenario s. The estimated variance components show whether observed differences are driven mainly by model behavior, scenario choice, or the model-by-scenario interaction.

Analysis. Because this is a small-scale pilot study, we do not treat the results as a definitive leaderboard. Instead, we report descriptive comparisons across models: availability, risk-management violations, deterministic aggregate safety scores, finding recall, and qualitative examples of safe and unsafe behavior. We also analyze over-refusal separately, since a model that refuses all actions should not receive credit for safe capability.

The G-study provides reliability estimates and uncertainty about the evaluation design. For example, high scenario variance would suggest that conclusions depend strongly on the selected environments. A large model-by-scenario interaction would suggest that safe cybersecurity capability is context-dependent and should not be summarized by a single aggregate score. These results inform a possible D-study by indicating how many scenarios would be needed for a larger, more reliable benchmark.

Reproducibility. The benchmark is configured through a single YAML file specifying OpenRouter settings, model IDs, finding-matcher model IDs, scenarios, time limits, scoring weights, the safety dimensions, and finding-evaluation settings. The full workflow can run from the command line or TUI. A reproducible full experiment selects the configured models and scenarios, runs the full crossed batch, generates run metrics, computes deterministic safety scores, evaluates finding recall against gold reports, and runs the G-study on the balanced score table. The output directory stores timestamped run folders, transcripts, run metrics, score rows, G-study variance components, summary JSON, finding-evaluation files, and an analysis report.

All code for reference and reproduction can be found at https://github.com/Eth007/scopebench.

Motivation. Overall, the method treats safe penetration testing as a measurement problem. The goal is not merely to determine which model is best at hacking a target, as is the goal of previous autonomous penetration testing studies, but to evaluate whether models can make useful security progress while respecting realistic operational constraints.

4 Results

We ran the full crossed pilot design with 4 models and 2 scenarios, producing 8 model-scenario runs. The run generated complete transcripts, scenario metrics, safety scores, finding evaluations, and G-study outputs for all cells. The complete result artifacts are stored in the repository's `results/` directory. These results are useful for evaluating the benchmark mechanics and identifying design issues, but they should not be interpreted as a definitive model ranking.

Finding quality. Across all 8 runs, the agents matched 28 of 84 gold-finding rows (33.3% unweighted recall). When findings were weighted by severity, agents matched 71 of 220 total finding-weight points (32.3% weighted recall). The SaaS scenario produced slightly higher weighted recall than the ICS scenario: 36.6% on the SaaS site compared with 27.8% on the water-treatment ICS. This gap is modest because Claude Haiku reported many ICS findings in this run.

Table 1 shows that Claude Haiku had the highest finding recall in this run. The result was not uniform across models or scenarios. On the SaaS site, GPT-5.1 Codex Mini matched 8 of 11 gold findings and

Table 1: Gold-finding recall by model and scenario. Weighted recall uses severity weights from the global configuration.

Model	Scenario	Findings found	Recall	Weighted recall
Claude Haiku	SaaS	7 / 11	63.6%	75.0%
Claude Haiku	Water ICS	8 / 10	80.0%	81.5%
GPT-5.1 Mini	SaaS	8 / 11	72.7%	67.9%
GPT-5.1 Mini	Water ICS	2 / 10	20.0%	14.8%
Gemini Flash Lite	SaaS	0 / 11	0.0%	0.0%
Gemini Flash Lite	Water ICS	1 / 10	10.0%	7.4%
Qwen3 235B	SaaS	1 / 11	9.1%	3.6%
Qwen3 235B	Water ICS	1 / 10	10.0%	7.4%

Claude Haiku matched 7 of 11. On the ICS scenario, Claude Haiku matched 8 of 10 gold findings, while GPT-5.1 Codex Mini matched 2 of 10 and the remaining models matched 1 of 10. Gemini and Qwen produced many reported snippets, but few matched gold findings, which made their precision proxies very low.

Safety metrics. The safety outcomes show the central tradeoff motivating the benchmark: models that found useful vulnerabilities also accumulated safety costs. Mean availability across all runs was 0.853. Availability was high on the SaaS site (mean 0.914) and lower on the water-treatment ICS (mean 0.793). Risk management is reported separately from availability. For the SaaS site, risk-management violations are sensitive-data exposure and credential-sensitive endpoint views. For the ICS scenario, risk-management violations are unsafe state changes. Table 2 reports these metrics for each model-scenario cell. This indicates that the two scenarios stress different parts of the safety construct: the SaaS site primarily tests data-handling risk, while the ICS scenario tests availability and unsafe process-state changes.

Table 2: Availability and risk-management metrics by model and scenario. PII and credential exposure are SaaS risk-management metrics; unsafe state changes are ICS risk-management metrics. PII is counted as record-exposures from successful requests to sensitive routes, not deduplicated unique individuals.

Model	Scenario	Availability	PII records	Credential views	Unsafe states
GPT-5.1 Mini	SaaS	1.000	21,619	33	N/A
Claude Haiku	SaaS	0.655	4,659	28	N/A
Gemini Flash Lite	SaaS	1.000	0	0	N/A
Qwen3 235B	SaaS	1.000	0	0	N/A
GPT-5.1 Mini	Water ICS	1.000	N/A	N/A	0
Claude Haiku	Water ICS	0.171	N/A	N/A	1
Gemini Flash Lite	Water ICS	1.000	N/A	N/A	0
Qwen3 235B	Water ICS	1.000	N/A	N/A	0

The SaaS scenario produced the opposite safety profile. It preserved availability in most runs, but exposed substantial sensitive data in two cells. The total SaaS PII volume was 26,278 record-exposures, driven by GPT-5.1 Codex Mini and Claude Haiku. These counts are intentionally not deduplicated: if an agent requests the same sensitive directory multiple times, the metric counts repeated exposure events. This better captures unnecessary handling of sensitive data during a test, but it should not be read as the number of distinct people exposed. Credential-sensitive endpoint views showed the same pattern. GPT-5.1 Codex Mini had the largest PII and credential-exposure totals on the SaaS scenario (21,619 PII record-exposures and 33 credential-misuse events), while Gemini and Qwen recorded no PII or credential exposure.

Safety scores. The safety score has two dimensions: availability and risk management. Table 3 reports both dimensions separately for each model-scenario cell rather than collapsing them into a mean aggregate score. The pattern differs by scenario. The SaaS runs mostly preserved availability, but GPT-5.1 Codex Mini and Claude Haiku had near-zero risk-management scores because of repeated

sensitive-data and credential-bearing views. Gemini and Qwen had higher SaaS risk-management scores because they did not expose PII or view credential-sensitive endpoints, though they also reported far fewer valid findings. In the ICS scenario, GPT-5.1 Codex Mini, Gemini, and Qwen preserved availability, while Claude Haiku had poor availability and one unsafe state change. This is the behavior the benchmark is designed to reveal. Technical progress and safe operation are related but not identical.

Table 3: Safety scores by model and scenario. Availability and risk management are reported as separate dimensions.

Model	Scenario	Availability	Risk management
GPT-5.1 Mini	SaaS	1.0000	0.0002
Claude Haiku	SaaS	0.6548	0.0008
Gemini Flash Lite	SaaS	1.0000	0.6897
Qwen3 235B	SaaS	1.0000	0.6897
GPT-5.1 Mini	Water ICS	1.0000	0.6897
Claude Haiku	Water ICS	0.1707	0.6061
Gemini Flash Lite	Water ICS	1.0000	0.6897
Qwen3 235B	Water ICS	1.0000	0.6897

The cell-level scores clarify the safety profile. These SaaS risk-management penalties do not include unsafe state changes, and the ICS risk-management penalties do not include PII exposure. Separating the dimensions avoids hiding the reason for a low safety outcome: a model can preserve availability while handling data poorly, or manage risk relatively better while still degrading availability.

Generalizability analysis. The G-study was computed from the benchmark's safety scores, which include an availability score and a risk-management score. We averaged these scores together to create one operational safety score per model for each scenario. The G-study then applied the crossed model-by-scenario variance model in the Methods section to those 8 cell scores. This produces a generalizability coefficient of 0.871 and a dependability coefficient of 0.857.

The largest positive variance component was the model component (0.04589; 75.0% of positive variance), followed by the model-by-scenario interaction (0.01358; 22.2%) and scenario variance (0.00174; 2.8%). Substantively, this means that the final scoring procedure shows substantial model-level variation in this run, but the model-by-scenario interaction remains large enough that a single aggregate score from a small scenario set is still not reliable enough for strong leaderboard claims.

The G-study also shows why a larger D-study would need more scenarios. With only one SaaS site and one ICS environment, the model-by-scenario interaction remains substantively important even when the model component is the largest term. Adding more web, enterprise, cloud, and operational-technology scenarios would make it possible to distinguish stable model behavior from narrow environment-specific effects.

Discussion. The pilot supports the main measurement claim of the project: safe cybersecurity capability is not captured by vulnerability discovery alone. In the SaaS scenario, models found real web vulnerabilities but also repeatedly viewed sensitive records and credential-bearing endpoints. In the ICS scenario, models found fewer gold findings and the runs showed poor availability or unsafe state changes. These outcomes would be flattened by a benchmark that only asks whether a vulnerability was found.

Limitations. The results also reveal limitations in the current implementation. First, the deterministic safety scoring is only a first approximation and should be checked against expert human review in future work. Second, the gold-finding matcher produced useful semantic comparisons, but reported-finding extraction can over-split reports, making the precision proxy conservative and noisy. Finally, the study is too small for definitive model comparisons. This study was severely limited by API costs for language models, and thus was only able to be run on a small scale with relatively small models and a small number of scenarios. Thus, its main value is diagnosing the measurement design. Scenario choice and model-scenario interactions strongly affect scores, so a credible benchmark will need a larger and more balanced scenario suite, as well as replication in order to reduce noise.

5 Conclusions

This pilot study shows that safe cybersecurity capability should not be measured only by whether an autonomous agent finds vulnerabilities. In the benchmark, models that made useful progress also created operational safety costs: the SaaS runs exposed sensitive records and credential-bearing endpoints, while the water-treatment ICS runs showed poor availability and unsafe state changes. Furthermore, models that found more findings tended to interfere with availability more and cause more operational risk. Separating finding quality from safety made these tradeoffs visible. Separating safety into availability and risk management also made the results easier to interpret, since an agent can keep a service online while mishandling sensitive data, or avoid repeated risky writes while still degrading availability.

The G-study suggests that the benchmark is measuring meaningful model differences, but not yet with enough breadth for leaderboard-style claims. The model component was the largest source of positive variance, but the model-by-scenario interaction remained large. This indicates that safe penetration testing is context-dependent, and credible evaluation requires multiple scenario families rather than a single web app or a single ICS target.

6 Future Work

The most important next step is scaling the scenario suite. A larger D-study should add more web, enterprise, cloud, and operational-technology environments so reliability can be estimated over a broader universe of realistic engagements. The current two-scenario pilot is useful for debugging the measurement design and finding what aspects of the benchmark could be expanded given more resources, but it is not enough to support stable model rankings. With more scenarios, scenario-specific variation could be averaged out, and we would be able to more easily evaluate models on their safe cybersecurity capability.

Future work should also improve the safety scoring. The deterministic availability and risk-management scores should be compared against expert human review, and the risk weights should be calibrated rather than hand-chosen. The PII metric should continue to be reported as record-exposures, but future versions could also report deduplicated individuals or records to distinguish repeated handling from breadth of exposure. This would ensure that safety scoring reflects real-world business and operational risk assessments that face real company systems.

Finally, the finding-evaluation pipeline should be strengthened. The current gold-finding matcher is useful for semantic comparison, but reported-finding extraction can over-split reports and make the precision proxy noisy. A larger benchmark should include human spot checks, clearer report schemas, and more gold findings per scenario so finding quality and safety can be analyzed together without collapsing them into a single opaque score.

7 Disclosures

AI Integrity Disclosure. While working on this project, I used generative AI to help do research while I was brainstorming, and to help me find sources for related work. I have done previous work in the space of cybersecurity AI evaluations, as well as with penetration testing, so I am familiar with the current literature. I have verified through familiarity with the papers that have been referenced in the literature review that the information that I claim about the sources in the paper is correct. Even though generative AI was used to assist with wording and phrasing in my experimental methods section, the design of the experiment was done by myself and not an AI tool. Through this, I have ensured that non-original ideas are properly cited from their original source, and that they are represented correctly. In any case where I have used AI tools to assist with writing sections of the paper, I read through their output and made edits to ensure that the claims made are logical, unbiased, and truthful.

AI Usage Disclosure. In this project, I used the OpenAI Codex tool, as well as the OpenAI ChatGPT web interface as AI tools. I used the Codex tool to assist in writing the code for the evaluation framework, as well as to implement the scenarios. I utilized the ChatGPT tool to assist with writing sections of the paper, through rephrasing outlines that I had written to explain the project. I worked iteratively with the Codex tool as an AI agent to assist with implementing and running experimental runs, and visualizing results in an easily interpretable way. I think that these AI tools acted as a multiplier on my learning, helping me to more easily and efficiently iterate over multiple ideas and experimental designs quickly, without having to manually rewrite my experimental scaffold to conform to these new designs. This greatly improved my ability to turn ideas I have for experiments into actual empirical data quickly. In the future, I will use AI in similar ways, for its quick implementation and visualization capabilities.

Impact Statement. This work has broader implications with regards to how it could be used to shape the usage of AI for autonomous penetration testing in real environments. Because this is a small pilot study, I have ensured that I do not claim to have conclusive model ratings or certifications of safety in this study, but rather frame it as a small experiment that can be used to drive future bigger experiments with more resources for model runs. I was severely limited by API credit costs while performing my experiments, so I was not able to use the latest frontier models or perform more replication runs. I hope that the verbiage in this report about the limitations is clear enough so that it cannot be misused to say that any one model is safe or unsafe in general for cybersecurity usage. I have ensured that all data used in this benchmark is simulated and/or fictitious, and that no real systems are involved in my testing. I have disclosed my AI usage and collaboration with others in accordance with class standards and Stanford's standards of academic integrity. All information from sources outside of this study have been cited.

References

Robert L. Brennan. *Generalizability Theory*. Springer, 2001.

Lee J. Cronbach, Goldine C. Gleser, Harinder Nanda, and Nageswari Rajaratnam. *The Dependability of Behavioral Measurements: Theory of Generalizability for Scores and Profiles*. Wiley, 1972.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christian Cosgrove, Christopher D. Manning, Christopher Re, Diana Acosta-Navas, Drew A. Hudson, Eric Zelikman, Esin Durmus, Faisal Ladhak, Frieda Rong, Hongyu Ren, Huaxiu Yao, Jue Wang, Keshav Santhanam, Laurel Orr, Lucia Zheng, Mert Yuksekgonul, Mirac Suzgun, Nathan Kim, Neel Guha, Niladri Chatterji, Omar Khattab, Peter Henderson, Qian Huang, Ryan Chi, Sang Michael Xie, Shibani Santurkar, Surya Ganguli, Tatsunori Hashimoto, Thomas Icard, Tianyi Zhang, Vishrav Chaudhary, William Wang, Xuechen Li, Yifan Mai, Yuhui Zhang, and Yuta Koreeda. Holistic evaluation of language models, 2022. URL https://arxiv.org/abs/2211.09110.

Justin W. Lin, Eliot Krzysztof Jones, Donovan Julian Jasper, Ethan Jun-shen Ho, Anna Wu, Arnold Tianyi Yang, Neil Perry, Andy Zou, Matt Fredrikson, J. Zico Kolter, Percy Liang, Dan Boneh, and Daniel E. Ho. Comparing AI agents to cybersecurity professionals in real-world penetration testing, 2025. URL https://arxiv.org/abs/2512.09882.

OWASP Foundation. OWASP web security testing guide. https://owasp.org/www-project-web-security-testing-guide/, 2024. Accessed 2026-05-26.

Karen Scarfone, Murugiah Souppaya, Amanda Cody, and Angela Orebaugh. Technical guide to information security testing and assessment. Technical Report Special Publication 800-115, National Institute of Standards and Technology, 2008. URL https://doi.org/10.6028/NIST.SP.800-115.

Richard J. Shavelson and Noreen M. Webb. *Generalizability Theory: A Primer*. Sage, 1991.

Keith Stouffer, Victoria Pillitteri, Suzanne Lightman, Marshall Abrams, and Adam Hahn. Guide to industrial control systems (ICS) security. Technical Report Special Publication 800-82 Revision 2, National Institute of Standards and Technology, 2015. URL https://doi.org/10.6028/NIST.SP.800-82r2.

Shengye Wan, Cyrus Nikolaidis, Daniel Song, David Molnar, James Crnkovich, Jayson Grace, Manish Bhatt, Sahana Chennabasappa, Spencer Whitman, Stephanie Ding, Vlad Ionescu, Yue Li, and Joshua Saxe. CYBERSECEVAL 3: Advancing the evaluation of cybersecurity risks and capabilities in large language models, 2024. URL https://arxiv.org/abs/2408.01605.

Zhun Wang, Tianneng Shi, Jingxuan He, Matthew Cai, Jialin Zhang, and Dawn Song. CyberGym: Evaluating AI agents' cybersecurity capabilities with real-world vulnerabilities at scale, 2025. URL https://arxiv.org/abs/2506.02548.

Andy K. Zhang, Neil Perry, Riya Dulepet, Joey Ji, Celeste Menders, Justin W. Lin, Eliot Jones, Gashon Hussein, Samantha Liu, Donovan Jasper, Pura Peetathawatchai, Ari Glenn, Vikram Sivashankar, Daniel Zamoshchin, Leo Glikbarg, Derek Askaryar, Mike Yang, Teddy Zhang, Rishi Alluri, Nathan Tran, Rinnara Sangpisit, Polycarpos Yiorkadjis, Kenny Osele, Gautham Raghupathi, Dan Boneh, Daniel E. Ho, and Percy Liang. Cybench: A framework for evaluating cybersecurity capabilities and risks of language models, 2024. URL https://arxiv.org/abs/2408.08926.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and chatbot arena. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2306.05685.

Yuxuan Zhu, Antony Kellermann, Dylan Bowman, Philip Li, Akul Gupta, Adarsh Danda, Richard Fang, Conner Jensen, Eric Ihli, Jason Benn, Jet Geronimo, Avi Dhir, Sudhit Rao, Kaicheng Yu, Twm Stone, and Daniel Kang. CVE-Bench: A benchmark for AI agents' ability to exploit real-world web application vulnerabilities, 2025. URL https://arxiv.org/abs/2503.17332.