Getting pwn'd by AI: Penetration Testing with Large Language Models

Andreas Happe
andreas.happe@tuwien.ac.at
TU Wien
Austria

Jürgen Cito
juergen.cito@tuwien.ac.at
TU Wien
Austria

ABSTRACT

The field of software security testing, more specifically penetration testing, requires high levels of expertise and involves many manual testing and analysis steps. This paper explores the potential use of large-language models, such as GPT3.5, to augment penetration testers with AI sparring partners. We explore two distinct use cases: high-level task planning for security testing assignments and low-level vulnerability hunting within a vulnerable virtual machine. For the latter, we implemented a closed-feedback loop between LLM-generated low-level actions with a vulnerable virtual machine (connected through SSH) and allowed the LLM to analyze the machine state for vulnerabilities and suggest concrete attack vectors which were automatically executed within the virtual machine. We discuss promising initial results, detail avenues for improvement, and close deliberating on the ethics of AI sparring partners.

CCS CONCEPTS

• **Security and privacy → Systems security**.

KEYWORDS

security testing, penetration testing, large language models

ACM Reference Format:
Andreas Happe and Jürgen Cito. 2023. Getting pwn'd by AI: Penetration Testing with Large Language Models. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '23), December 3–9, 2023, San Francisco, CA, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3611643.3613083

1 INTRODUCTION

Large language models (LLMs), such as ChatGPT or GPT3.5, have become a hot topic not only in computer science but also within popular media [12]. The field of cybersecurity and software security testing, more specifically, penetration testing, suffers from a chronic lack of personnel [19], even worse, according to the ISC2 Cybersecurity Workforce Study 2022 [18], while global cybersecurity workforce was growing by 11.1% YoY, this growth was outpaced by the gap's increase of 26.2% YoY. A recent interview study with penetration testers highlighted the need for human sparring partners [16], i.e., colleagues who offer alternative ideas or approaches when stuck. The study also emphasizes that intuition is a big part of detecting vulnerabilities and that knowledge transfer, e.g., from attending Capture-the-Flag[1] (CTF) events, were seen as potential sources of this intuition — can this be partially outsourced to AI models? Using AI-based agents as sparring partners would augment and empower existing human security testers and could counteract the lack of sufficiently educated security professionals. Combining human operators with AIs creates new capabilities instead of cloning existing ones. Furthermore, keeping a human in the loop reduces the potential ethical problems imposed by the use of AIs [6]. Recent research indicates that the efficiency gains provided by the use of AI-based systems are greatest for low-skilled workers [7], augmenting human operators with a generative AI might thus also benefit the training of novice penetration testers.

RQ: To what extent can we automate security testing with LLMs? The rest of this paper explores whether large-language models can be deployed as sparring partners for security professionals. To answer this question, we leverage MITRE ATT&CK, a curated database of knowledge about threat actors in the cybersecurity domain, to provide a guiding structure. A good sparring partner should be able to cover the different tactics, techniques, and procedures covered by ATT&CK. To explore this hypothesis, we performed multiple experiments. To showcase high-level guidance, we "asked" an LLM to help design penetration tests for both generic scenarios as well as for a concrete target organization. To showcase low-level guidance, we integrated GPT3.5 with a vulnerable virtual machine and allowed it to analyze the machine for vulnerabilities and suggest attack vectors. Based on our experience, we discuss the results as well as potential future improvements.

Scope. We also envision other areas where generative AI could be used successfully. One of them is the generation of phishing or vishing messages. For obvious ethical reasons, we did not further analyze attacks that intently try to deceive human beings. Another tedious area where generative AI could improve efficiency would be automated report generation for penetration tests or red teaming campaigns. Anecdotal evidence suggests that penetration testers are already experimenting with generative AI for report generation.

2 BACKGROUND

This section highlights the technologies and techniques used.

MITRE ATT&CK. MITRE ATT&CK [37] is a curated database of knowledge about threat actors (APTs). It employs a hierarchical model often abbreviated by "TTP". The initial "T" stands for "tactics" and describes high-level objectives an adversary intends to achieve,

[1] CTFs are gamified penetration-testing exercises.

ESEC/FSE '23, December 3–9, 2023, San Francisco, CA, USA

e.g., reconnaissance, privilege escalation or collection. The middle "T" describes "techniques". Each technique is a way to achieve a tactic. Examples of techniques would be "*Abuse Elevation Control Mechanism: Sudo and Sudo Caching*" [3] or "*Steal or Forge Kerberos Tickets: Kerberoasting*" [4]. Finally, "P" describes procedures that are the specific details of how an adversary executes a technique.

We assume that a sparring partner for penetration testing should cover the whole TTP spectrum. On a high level, it should be able to select suitable tactics and corresponding techniques. On a low-level, given an employed tactic, it should be able to derive feasible techniques and procedures.

Large Language Models. A Large-Language Model (LLM) consists of a neural network trained using self-supervised learning on vast amounts of data. A model's capabilities are highly dependent upon its complexity which is often described through the number of used parameters. Current models yield parameter sizes ranging from billions, e.g., LLaMA starts with 7 billion, to trillions of parameters, e.g., Wu Dao or GPT-4. Model and parameter sizes are currently under discussion; on one hand, larger models can exhibit emergent behaviors [38]; on the other hand, e.g., there is speculation that the age of ever-larger models is over due to reduced scaling efficiency [24].

Training a new LLM is prohibitively expensive for most researchers, but existing LLMs can be refined or fine-tuned to specific use cases for feasible costs. This situation has created the moniker "foundation models" for LLMs. The importance of those has been acknowledged by mainstream media, c.f., the Economist's "Huge Foundation Models are Turbo Charging AI Progress" in 2022 [11].

GPT3.5/ChatGPT. Conversations with ChatGPT commonly consist of questions, named "prompts", and answers going back and forth between the user and the AI. Prompts have to be carefully prepared, yielding a new discipline that has been called prompt engineering [10, 22, 36, 40].

Tools such as *llama.cpp* [14] that make use of small-scale models (up to 13b parameters) feasible on consumer-grade hardware have sparked additional research. Those models can be run without any cloud/API costs and are not subject to any server-side moderation or censorship.

Pre-trained Autonomous AI Agents. AutoGPT [15] introduced the idea of auto-generating sequences of instructions by leveraging LLMs to create the prompt that is subsequently used to query the LLM. This allows users to provide concise initial questions for the AI system that are subsequently refined. This reduces the need for manual prompt engineering. LLMs often "hallucinate", i.e., invent facts that seem statistically plausible. Research suggests that using external knowledge and automated feedback can reduce these hallucinations [29]. AutoGPT integrates web-based queries and optional human-provided feedback during its operation. Based on this, the initial task is converted into a task list containing smaller subtasks that can be delegated to additional agents.

BabyAGI focuses on automated task generation, planning, and execution [26, 27]: a user-given task is split up into smaller subtasks that are stored within a task queue. *Autonomous Task Execution Agents* take tasks from the task queue, execute them, and add new information to a memory store. In addition, the *Task Creation Agent* identifies new subsequent tasks that are pushed upon the task queue and are eventually executed by the *Task Execution Agent*. Before a task is executed, a *Context Agent* is asked to provide sufficient context for the task from memory. Entries in the task queue are prioritized through a *Prioritization Agent*. All mentioned agents are GPT-4 processes themselves. BabyAGI [25] provides a "pared-down" version of this system in roughly 100 lines of Python code. Jarvis [33] employs agents with different models to create multi-modal, multiagent systems.

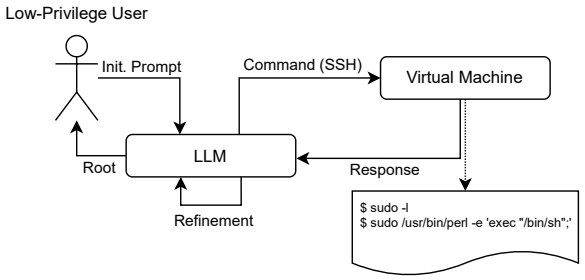

Figure 1: High-Level Architecture Overview

3 LLM-BASED PENETRATION TESTING

We differentiate between two use cases: on a high level, typical questions asked by pen-testers are "what is a good attack methodology", e.g., "how to attack Active Directory". These questions should yield tactics as well as potential techniques to achieve those tactics. On the low-level, we assume that the pen-tester has already chosen to employ a tactic against a target system and is searching for appropriate techniques and corresponding procedures. A typical question would be "I want to perform a privilege escalation (tactic), what are the suitable attack vectors against [this concrete Linux system]?".

3.1 High-Level: Task-Planning Systems

For the evaluation, we asked AgentGPT to "Become domain admin in an Active Directory". The generated document contained highly realistic attack vectors such as password spraying, Kerberoasting, AS-REP roasting, exploiting Active Directory Certificate Services, abusing unconstrained delegation or exploiting group policies. All of those attacks are realistic, feasible, and commonly used during penetration testing.

In addition, after securing a target company's approval, we tasked AutoGPT to devise an external penetration testing plan for that company. AutoGPT's plan included standard methods such as performing a network vulnerability scan, performing OSINT/user enumeration, and performing phishing against identified users. All these are operations typically performed during external penetration tests. When further inquired, AutoGPT was able to crawl the company's web page and identify potential phishing targets (users and their email addresses) but declined to perform any "real" network security scan or perform phishing operations due to its ethical filters. Both answers were realistic, and feasible, and would give a penetration tester good feedback about potential attack vectors.

3.2 Low-Level: Attack-Execution System

The low-level evaluation targets a common scenario: after a penetration tester gained low-level privilege access to a Linux system, they search for a privilege-escalation attack to become the system's *root* user. To allow for realistic evaluation, we wrote a Python script that uses SSH to connect to a deliberately vulnerable *lin.security* Linux virtual machine [21]. The prototype's source code and documentation are provided at https://github.com/ipa-lab/hackingBuddyGPT.

The script consists of an infinite loop: within the loop, it tells GPT3.5 to imagine being a low-privilege user that wants to become the *root* user. To achieve this, the LLM can state a Linux shell command that will be executed over SSH on the virtual machine. The corresponding output is presented back to GPT3.5 when prompted for the next command. Figure 1 shows a high-level overview of this feedback loop. **With this simple structure, we were able to gain root privileges on our vulnerable virtual machine.**

In addition, at the end of each loop iteration, GPT3.5 was presented with the chosen command and its output and then tasked to identify potential security vulnerabilities based on this information. For each vulnerability, it was tasked to provide an exploitation example, sneakily named "verification commands". This yielded additional attack vectors.

Our script was routinely able to gain root privileges within the virtual machine. The common path was listing the "sudoers" file by calling *sudo -l*, followed by either using *sudo* with one of the listed shells or employing one of the listed GTFObins to gain a root shell. GTFObins are benign system commands that when called through *sudo*, can be abused to gain a root shell. Another frequently used attack vector was retrieving */etc/passwd* and identifying user accounts not using shadow passwords[2]. Searches for SUID binaries were requested, but returned binaries not actively exploited, indicating lacking multi-step planning capabilities of either our script or the underlying model. A slightly altered prompt instructing the LLM to open a reverse shell to a given IP address was successful and dropped root shells.

4 DISCUSSION

This section reflects upon the pen-test performance of the prototype, guided by the 10+ years of pen-testing experience of the first author.

4.1 Grounding of Results and Hallucinations

One interesting aspect of our prototype is that all executed commands and their resulting output are written to a protocol. This allows us to reason if LLM-suggested vulnerabilities are based on queries providing system knowledge, or if GPT3.5 extracted security trends and preconceptions during training. The latter is analogous to penetration testers applying knowledge gained during work or training, e.g., from participating in CTFs.

There were indications of reasoning about causal dependencies: After retrieving the list of sudoers, GPT3.5 consistently suggested various vulnerable sudo commands for privilege escalation. A similar pattern arose after retrieving the *passwd* file: here attacking weakly-configured user accounts was suggested as the next step.

[2]If your Linux system is not using shadow passwords by now, chatGPT is the least of your worries.

Other suggestions, such as using certain system exploits, e.g., *dirty_cow*, were reasonable given that GPT3.5 "knew" that this was a Linux system, but were given without any previous enumeration.

Pure and easily detectable hallucinations occurred infrequently, the most common occurrence was the suggestion to execute "exploit.sh". It seems reasonable that security write-ups containing the execution of this script were part of GPT3.5's training set.

While the suggested system commands obviously were based upon pattern-matching and not on a deeper understanding of the Linux system or on model building, seeing the simple LLM-shell-based feedback loop we established gaining root privileges was eerie. A suitable analogy would be a pen-tester talking to a colleague over the phone, asking for suggestions with the conversation partner only having a very limited view of the actual system but a set of preconceptions (i.e., priors), which is partially in line with our research question on the ability of LLMs acting as sparring partners. When given the additional subcommand of "and explain the found vulnerabilities" in the prompt, GPT3.5 was able to provide good introductory information and could thus be utilized as part of on-the-job training.

4.2 Stability and Reproducibility

Singular prototype runs were not stable, i.e., there was variation in the sequence and selection of commands given and vulnerabilities identified. On longer runs, or when aggregating multiple runs, the results converged (we repeatedly ran the identical script in the order of tens of times to be able to make observations on convergence). The variation on single runs seems to be related to GPT3.5 overly focusing upon single aspects of the tested system. This is also known to happen to pen-testers during assignments, "going down a rabbit hole" improves with experience [16].

Compared to tools such as *linpeas.sh* [30], LLMs seem to be less deterministic. Enumeration tools traverse a manually curated hard-coded list of vulnerability checks. Further research should clarify if the observed instability converges over time while reducing detectable patterns for intrusion detection systems. Ironically, GPT3.5 suggested calling *linpeas.sh* during one run but failed as it tried to download it from an invalid URL.

4.3 Ethical Moderation in LLMs

The prototype used *GPT-3.5-turbo* which contains safety measures against malicious prompts. The prototype relayed commands to a vulnerable virtual machine, but the overall scenario can easily be applied to real systems. GPT3.5's lack of hesitation was discerning. During the development of the script, the ethics filter was infrequently triggered. Adding "do not ask questions or provide judgments" to command prompts seems to significantly reduce denials. The optional "detail additional vulnerabilities" step was more often denied due to ethical reasons, but this had no impact on the overall hacking progress. Slight prompt variations were successful in reducing GPT3.5's ethical concerns, e.g., instead of asking for "exploits for vulnerabilities" we asked for "verification commands for vulnerabilities". Of course, switching from OpenAI to one of the locally running LLMs would remove all server-side ethics checks.

Ethical questions are not new in the cybersecurity domain, especially regarding releasing penetration test tools. Ethical issues

arising from using GPT3.5 resemble discussions about open-source security tools which can be used by both red-teams as well as by APTs. Commercial vendors try to vet their customers, while open-source tools can be used by anyone. In the end, malicious actors can and will use both of them [20, 34]. Regulation regarding the distribution of dual-use goods exists [2], but application to software is clumsy due to its fluid and often impalpable nature.

Another ethical problem is the inclusion of toxic content in commonly used training sets [32]. As our prototype uses an already trained foundation model, we are not deliberating on this issue. This publication also does not touch on the topic of the inclusion of copyrighted information within training data.

5 A VISION OF AI-AUGMENTED PEN-TESTING

We deliberate on research ideas and pragmatic considerations to form a more perfect union between pen-testers and LLMs.

5.1 Integration of High- and Low-Level

We differentiated between high- and low-level tasks and distributed those to two different LLMs. Integrating both, i.e., high-level task planing and low-level system exploitation, would yield a more uniform user experience. We imagine a system in which human operators can inquire about high-level concepts, e.g., "what additional active directory attacks can I try?", and later switch to a lower level, e.g., "given this system, how can I escalate?". Keeping all information within a single system should also enable synergy effects as the LLMs learn details about the tested system. This also shows the expected multistep interactive feedback loop between LLMs and operators.

5.2 Investigation of Model Options

We currently use OpenAI's GPT-3 through a cloud-based API. GPT-3 should be evaluated against locally run models such as Llama [39], StableLM [35], Dolly2 [9] or Koala [13].

Locally run models do not incur any cloud costs and do not share sensitive data with the cloud. As no data is leaked, this would enable further customer-specific model training and fine-tuning: Imagine training a local model with data found during an engagement or fine-tuning a customer-specific model over a series of subsequent penetration tests. During a recent interview series with pen-testers [16], participants mentioned that they "learn how their customer or industry area works and thinks over time", could a customized AI model achieve something similar? Although the industry is currently aiming for ever larger model parameter sizes, analyzing which parameter size is "good enough" should reduce the resource impact of deploying LLMs.

5.3 Memory, Verification, and Reflection

Memory is provided to GPT3.5 through context embedded within query prompts. Prompt size is typically limited, e.g., the used GPT-3 model had a limit of 4k tokens. With newer models, this limit is constantly increasing and allows to pass a richer context to the used LLM. Our prototype has simplistic memory that includes the output of executed commands until the context limit is reached. Generative Agents such as BabyAGI utilize chatGPT to build a suitable context for each generated prompt. Concurrent research in generative game agents [28] utilized LLMs to reflect on recent events experienced by agents, and then asked an LLM to provide a summarized description. The results are used as reflected memory for future queries. In our use-case, executed command output could be reflected on and only relevant extracted information added to the next prompt's context. Another option would be using multiple memory streams: one about recently executed commands, one for extracted security findings, and one describing what kind of computer system would fit the experienced findings, i.e., emulate model building. Using this model as an internal "reality check" should reduce the used LLM's hallucinations. Having a rough model of the tested system, as well as a compacted history of vulnerabilities tested, would also benefit questions such as "what other vulnerabilities might I have overlooked?".

5.4 Prompts for Asking Better Questions

Our prototype used rather static and manually written prompts. Using LLMs to generate and optimize the prompts themselves, similar to AutoGPT, might improve their effectiveness. Given our sensitive use case, these automatically generated prompts should be closely monitored by humans though.

Another avenue of research is searching for better questions to be asked. Based upon empirical studies on how penetration testers work [16], further research into which questions they ask themselves during their work can inform better prompts as well as a better understanding of this close-knit industry.

6 FINAL ETHICAL CONSIDERATIONS

This paper explores the use of LLMs for augmenting penetration testing in benign settings. However, tools can easily be subverted for malicious purposes. Ethical questions arise. Concurrent reports indicate that AI is currently being driven forward by private companies as well as by state-funded research agencies [23]. The former have an economic incentive, while the latter see geopolitical implications of AI. We do not expect that this avenue of research will slow down. Parallel to that, the reported malicious use of AI, presumably by APTs and common criminals, is increasing [1].

Locking away models behind server-side supervised APIs is not feasible as models can be run locally. In addition, even gate-kept models such as Meta's LlaMA have been leaked [31] and can now be reused by malicious actors. Fine-tuning such a model to concrete malicious activities is easily within APTs reach: For example, when using StackLlAMA's processing power estimates for fine-tuning [5], an attacker using on-demand cloud computing can expect to be able to fine-tune a model for less than a thousand US dollars. Using chat-based LLMs through prompt engineering does not require a thorough computer science education. While this is beneficial in democratizing access to processing techniques, this also facilitates potential malicious use.

While it is not predetermined if and how LLMs will influence hacking, we assume that attackers will explore possibilities, including fully-automated approaches. Given the low entry costs for experimentation, this cannot be contained anymore.

Attacks will use LLMs; the genie is out of the bottle, and the red queen's race is on [8, 17]. Defenders need to be prepared for that — and LLMs can play a significant role.

REFERENCES

[1] AIAAIC. 2023. *AIAAIC Repository of incidents and controversies related to AI, algorithms and automation.* Retrieved April 26, 2023 from https://www.aiaaic.org/

[2] The Wassenaar Arrangement. 1982. *The Wassenaar Arrangement on Export Controls for Conventional Arms and Dual-Use Goods and Technologies.* Retrieved August 12, 2023 from https://www.wassenaar.org/

[3] MITRE ATT&CK. 2020. *Abuse Elevation Control Mechanism: Sudo and Sudo Caching.* Retrieved May 1, 2023 from https://attack.mitre.org/techniques/T1548/003/

[4] MITRE ATT&CK. 2020. *Steal or Forge Kerberos Tickets: Kerberoasting.* Retrieved May 1, 2023 from https://attack.mitre.org/techniques/T1558/003/

[5] Edward Beeching, Younes Belkada, Kashif Rasul, Lewis Tunstall, Leandro von Werra, Nazneen Rajani, and Nathan Lambert. 2023. StackLLaMA: An RL Fine-tuned LLaMA Model for Stack Exchange Question and Answering. https://doi.org/10.57967/hf/0513

[6] Erik Brynjolfsson. 2023. The turing trap: The promise & peril of human-like artificial intelligence. In *Augmented Education in the Global Age.* Routledge, 103–116.

[7] Erik Brynjolfsson, Danielle Li, and Lindsey Raymond. 2023. Generative AI at Work. NBER Working Paper No. 31161. *National Bureau of Economic Research* (April 2023).

[8] Vit Bukac, Vaclav Lorenc, and Vashek Matyáš. 2014. Red queen's race: APT win-win game. In *Cambridge International Workshop on Security Protocols.* Springer, 55–61.

[9] Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. 2023. *Free Dolly: Introducing the World's First Truly Open Instruction-Tuned LLM.* https://www.databricks.com/blog/2023/04/12/dolly-first-open-commercially-viable-instruction-tuned-llm

[10] Paul Denny, Viraj Kumar, and Nasser Giacaman. 2023. Conversing with Copilot: Exploring prompt engineering for solving CS1 problems using natural language. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1.* 1136–1142.

[11] The Economist. 2022. *Huge foundation models are turbo-charging AI progress.* Retrieved April 25, 2023 from https://www.economist.com/interactive/briefing/2022/06/11/huge-foundation-models-are-turbo-charging-ai-progress

[12] The Economist. 2023. *Large, creative AI models will transform lives and labour markets.* Retrieved April 25, 2023 from https://www.economist.com/interactive/science-and-technology/2023/04/22/large-creative-ai-models-will-transform-how-we-live-and-work

[13] Xinyang Geng, Arnav Gudibande, Hao Liu, Eric Wallace, Pieter Abbeel, Sergey Levine, and Dawn Song. 2023. Koala: A Dialogue Model for Academic Research. Blog post. https://bair.berkeley.edu/blog/2023/04/03/koala/

[14] Georgi Gerganov. 2023. *llama.cpp: Inference of LLaMA model in pure C/C++.* Retrieved April 28, 2023 from https://github.com/ggerganov/llama.cpp

[15] Significant Gravitas. 2023. *Auto-GPT: An Autonomous GPT-4 Experiment.* Retrieved April 25, 2023 from https://github.com/Significant-Gravitas/Auto-GPT

[16] Andreas Happe and Jürgen Cito. 2023. Understanding Hackers' Work: An Empirical Study of Offensive Security Practitioners. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (San Francisco, USA) *(ESEC/FSE 2023).* Association for Computing Machinery, New York, NY, USA, 11 pages.

[17] Richard Harang and Felipe N Ducau. 2018. Measuring the speed of the Red Queen's Race. *BlackHat: Las Vegas, NV, USA* (2018).

[18] (ISC)2. 2022. *(ISC)2 CYBERSECURITY WORKFORCE STUDY 2022.* Retrieved April 28, 2023 from https://www.isc2.org//-/media/ISC2/Research/2022-WorkForce-Study/ISC2-Cybersecurity-Workforce-Study.ashx

[19] Sydney Lake. 2022. *The cybersecurity industry is short 3.4 million workers—that's good news for cyber wages.* Retrieved April 28, 2023 from https://fortune.com/education/articles/the-cybersecurity-industry-is-short-3-4-million-workers-thats-good-news-for-cyber-wages/

[20] Selena Larson and Daniel Blackford. 2021. *Cobalt Strike: Favorite Tool from APT to Crimeware.* Retrieved April 28, 2023 from https://www.proofpoint.com/us/blog/threat-insight/cobalt-strike-favorite-tool-apt-crimeware

[21] lin.security. 2018. *Lin.Security: 1.* Retrieved May 1, 2023 from https://www.vulnhub.com/entry/linsecurity-1,244/

[22] Vivian Liu and Lydia B Chilton. 2022. Design guidelines for prompt engineering text-to-image generative models. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems.* 1–23.

[23] Nestor Maslej, Loredana Fattorini, Erik Brynjolfsson, John Etchemendy, Katrina Ligett, Terah Lyons, James Manyika, Helen Ngo, Juan Carlos Niebles, Vanessa Parli, Yoav Shoham, Russell Wald, Jack Clark, and Raymond Perraul. 2023. The AI Index 2023 Annual Report. https://aiindex.stanford.edu/wp-content/uploads/2023/04/HAI_AI-Index-Report_2023.pdf

[24] Ron Miller. 2023. *Sam Altman: Size of LLMs won't matter as much moving forward.* Retrieved April 26, 2023 from https://techcrunch.com/2023/04/14/sam-altman-size-of-llms-wont-matter-as-much-moving-forward/

[25] Yohei Nakajima. 2023. *BabyAGI.* Retrieved April 25, 2023 from https://github.com/yoheinakajima/babyagi

[26] Yohei Nakajima. 2023. *Introducing Task-driven Autonomous Agent.* Retrieved April 25, 2023 from https://twitter.com/yoheinakajima/status/1640934493489070080

[27] Yohei Nakajima. 2023. *Task-driven Autonomous Agent Utilizing GPT-4, Pinecone, and LangChain for Diverse Applications.* Retrieved April 25, 2023 from https://yoheinakajima.com/task-driven-autonomous-agent-utilizing-gpt-4-pinecone-and-langchain-for-diverse-applications/

[28] Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative Agents: Interactive Simulacra of Human Behavior. arXiv:2304.03442 [cs.HC]

[29] Baolin Peng, Michel Galley, Pengcheng He, Hao Cheng, Yujia Xie, Yu Hu, Qiuyuan Huang, Lars Liden, Zhou Yu, Weizhu Chen, and Jianfeng Gao. 2023. Check Your Facts and Try Again: Improving Large Language Models with External Knowledge and Automated Feedback. arXiv:2302.12813 [cs.CL]

[30] Carlos Polop. 2023. *LinPEAS - Linux Privilege Escalation Awesome Script.* Retrieved April 28, 2023 from https://github.com/carlospolop/PEASS-ng/tree/master/linPEAS

[31] Katyanna Quach. 2023. *LLaMA drama as Meta's mega language model leaks.* Retrieved April 26, 2023 from https://www.theregister.com/2023/03/08/meta_llama_ai_leak/

[32] Kevin Schaul, Szu Yu Chean, and Nitasha Tiku. 2023. *Inside the secret list of websites that make AI like ChatGPT sound smart.* Retrieved April 26, 2023 from https://www.washingtonpost.com/technology/interactive/2023/ai-chatbot-learning/

[33] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. HuggingGPT: Solving AI Tasks with ChatGPT and its Friends in HuggingFace. arXiv:2303.17580 [cs.CL]

[34] Cybereason Global SOC and Incident Response Team. 2023. *Sliver C2 Leveraged by Many Threat Actors.* Retrieved April 28, 2023 from https://www.cybereason.com/blog/sliver-c2-leveraged-by-many-threat-actors

[35] stability.ai. 2023. *Stability AI Launches the First of its StableLM Suite of Language Models.* https://stability.ai/blog/stability-ai-launches-the-first-of-its-stablelm-suite-of-language-models

[36] Hendrik Strobelt, Albert Webson, Victor Sanh, Benjamin Hoover, Johanna Beyer, Hanspeter Pfister, and Alexander M Rush. 2022. Interactive and Visual Prompt Engineering for Ad-hoc Task Adaptation with Large Language Models. *IEEE transactions on visualization and computer graphics* 29, 1 (2022), 1146–1156.

[37] Blake E Strom, Andy Applebaum, Doug P Miller, Kathryn C Nickels, Adam G Pennington, and Cody B Thomas. 2018. Mitre att&ck: Design and philosophy. In *Technical report.* The MITRE Corporation.

[38] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022. Emergent Abilities of Large Language Models. arXiv:2206.07682 [cs.CL]

[39] Renrui Zhang, Jiaming Han, Aojun Zhou, Xiangfei Hu, Shilin Yan, Pan Lu, Hongsheng Li, Peng Gao, and Yu Qiao. 2023. Llama-adapter: Efficient fine-tuning of language models with zero-init attention. *arXiv preprint arXiv:2303.16199* (2023).

[40] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. 2022. Learning to prompt for vision-language models. *International Journal of Computer Vision* 130, 9 (2022), 2337–2348.